| | |
|---|---|
| **Titre:** Title: | Modélisation d'un agent de recherche intelligente d'information sur internet |
| **Auteur:** Author: | Wassim Moussawi |
| **Date:** | 2000 |
| **Type:** | Mémoire ou thèse / Dissertation or Thesis |
| **Référence:** Citation: | Moussawi, W. (2000). Modélisation d'un agent de recherche intelligente d'information sur internet [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/8903/ |

UNIVERSITÉ DE MONTRÉAL

# MODÉLISATION D'UN AGENT DE RECHERCHE INTELLIGENTE D'INFORMATION SUR INTERNET

WASSIM MOUSSAWI

DÉPARTEMENT DE GÉNIE ÉLECTRIQUE ET DE GÉNIE INFORMATIQUE

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION

DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES

(GÉNIE ÉLECTRIQUE)

NOVEMBRE 2000

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

# MODÉLISATION D'UN AGENT DE RECHERCHE INTELLIGENTE D'INFORMATION SUR INTERNET

présenté par : MOUSSAWI Wassim

en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de :

M. BERNARD Jean-Charles, Ph.D., président

M. HOANG Hai-Hoc, Ph.D., membre et directeur de recherche

M. PIERRE Samuel, Ph.D., membre et codirecteur de recherche

M. PROBST Wilfried, Ph.D., membre (UQAM)

*À mes parents et à Céline. Sans votre amour, votre soutien, vos encouragements et votre souhait de me voir atteindre le but je n'aurai sans doute pas pu aujourd'hui accomplir ce mémoire...*

# REMERCIEMENTS

Le dépôt de ce mémoire n'aurait sans doute jamais vu le jour sans l'appui constant reçu de la part de Monsieur Samuel Pierre tout au long de ma recherche. La confiance et la détermination, dont il a fait preuve envers moi, furent infléchissables à travers le temps. Le contenu de la recherche a aussi considérablement bénéficié de la grande rigueur, des conseils judicieux et des relectures minutieuses, de Monsieur Haï Hoc Hoang. Mes connaissances et mon expérience se sont indéniablement élargies à travers toutes les discussions enrichissantes que j'ai eues avec mes directeurs, Hoang et Pierre, avec qui des liens conviviaux et amicaux se sont définitivement établis.

# RÉSUMÉ

Ces dernières années, la plus importante augmentation de stockage et de communication d'informations s'est produite sur Internet. L'apparition du Web a été à l'origine de la croissance exceptionnelle des communications et de la publication d'informations sur Internet. Cette croissance phénoménale de l'information publiée sur le Web, puisqu'elle entraîne une répartition et une dissémination de l'information, a lancé le grand défi de la recherche de données pertinentes dans ce vaste océan d'informations.

Les moteurs de recherche mis à la disposition des internautes à des fins de recherche d'informations sont certes des outils de base utiles, mais ils comportent des handicaps majeurs liés à leur utilisation. L'un des inconvénients majeurs inhérents aux moteurs de recherche est l'imprécision et la non pertinence des résultats. Qui n'a pas eu l'impression de perdre du temps en voulant trouver une information précise sur l'Internet à l'aide d'un moteur de recherche ? Bien souvent, à l'issue d'une requête formulée auprès d'un moteur de recherche, l'utilisateur se trouve submergé par un volume impressionnant de réponses non pertinentes. À ce stade, une revue manuelle de la part de l'utilisateur est souvent requise afin de trouver, parmi l'ensemble des réponses retournées, celles qui sont éventuellement pertinentes.

Les travaux présentés dans ce mémoire portent sur l'élaboration et l'implantation partielle d'un agent, nommé ARIII, pour la recherche intelligente d'information sur Internet. La fonction principale de cet agent est de fournir des services de recherche d'informations sur Internet et d'assister l'utilisateur lors de sa recherche. Les objectifs principaux de l'agent ARIII sont : établir un mécanisme de création et de manipulation de profil d'utilisateur autour duquel s'articule le processus de la recherche d'information ; simplifier la formulation de la requête, tout en rendant son contenu plus pertinent ; améliorer la pertinence des réponses et établir un classement des résultats basé sur le profil de l'utilisateur ; mettre en place des mécanismes de collecte de la

réaction de l'utilisateur face aux résultats, permettant ainsi de raffiner son profil ; soulager l'utilisateur en minimisant les interactions qui servent à diriger la recherche, en les remplaçant par la mise en place de mécanismes d'autonomie et de proactivité basés sur les informations présentes dans le profil de l'utilisateur.

Le modèle d'agent ARIII présenté dans le cadre de ce mémoire tente de remédier à certaines faiblesses inhérentes aux moteurs de recherche par index et de tirer profit des nombreux avantages présentés par la recherche d'information basée sur l'espace vectoriel. De plus, il apporte plusieurs contributions aux notion de profil d'utilisateur, de formulation de requête, de raffinement des requêtes et de filtrage et présentation des résultats, propres à la théorie de recherche d'information. Au niveau pratique, notre modèle propose l'utilisation du profil de l'utilisateur afin de soulager une formulation complexe des requêtes et de cibler les résultats pertinents aux besoins informationnels de l'utilisateur en question. Au niveau viabilité, l'implantation partielle de notre modèle, a donné lieu à un prototype qui a montré, lors d'une expérimentation comparative avec l'engin de recherche par index Altavista, une grande fiabilité et une pertinence améliorée dans le classement des résultats.

Finalement, puisqu'il est essentiellement lié au problème de recherche d'information, le modèle présenté contribue à ce domaine par la mise en place d'éléments permettant aux utilisateurs de définir leurs besoins et aux responsables du système de recherche d'information d'encadrer et de définir le contenu et l'étendue de l'information par l'entremise d'un mécanisme de collecte d'information à partir d'une liste embryonnaire de documents.

# ABSTRACT

Due to the past decade advances in the computer technologies, the Internet becomes the largest, ultimate and most used source of information. The Internet now seems to have an inexhaustible supply of information on every conceivable subject. To many, this is its major fascination, attraction, and strength. However, the opposite case is also true, in that the quantity of information available is a weakness. The sheer volume of information to search is almost unmanageable. The results returned from searches by conventional search engines often supply information that cannot possibly be analyzed in a reasonable time frame. For example, a search using a conventional search engine such Infoseek asked this simple, straightforward question: « Can you find me a job? » The search engine returned 39,746,521 pages. If we allow 30 seconds to review each page, it would take 37 years and 6 months, with no breaks, to review all the pages. This review would be a full-time occupation in itself! Often we are overwhelmed by these search results and so abandon them. If we look hard and long enough, we might find what we want, or at best, we compromise our search or the results we are seeking.

Thus, users need an intelligent and efficient information search tool that could relieve them from the hard task of reviewing a large amount of results. This thesis proposes an agent, named ARIII, which fulfills this purpose. The main function of this agent is to discover and retrieve on the behalf of the user pertinent information over the Internet by taking in consideration the user profile and performing a pertinent sort and filtering of the results that matches the user's preferences and interests.

# TABLE DES MATIÈRES

# LISTE DES FIGURES

# LISTE DES SIGLES ET ABBRÉVIATIONS

| Sigle ou Abréviation | Signification |
|---|---|
| AFNOR | Association Française de NORmalisation |
| ARIII | Agent de Recherche Intelligente d'Information sur Internet |
| EDI | Environnement de Développement Intégré |
| FIPA | Foundation for Intelligent Physical Agents |
| FTP | File Transfert Protocol |
| GUI | Graphic User Interface |
| HTML | Hyper Text Markup Language |
| IA | Intelligence Artificielle |
| ISAME | Information - Système d'Aide Multi-Experts |
| KIF | Knowledge Interchange Format |
| KQML | Knowledge Query Manipulation Language |
| LIRA | Learning Information Retrieval Agents |
| OMT | Object Modeling Technique |
| OOSE | Object Oriented Software Engineering |
| PDF | Printed Document Format |
| RMI | Remote Method Invocation |
| SA | Structured Analysis |
| SADT | Structured Analysis Design Technique |
| SART | Structured Analysis with Real Time extensions |
| SRII | Système de recherche intelligente d'information |
| TCP/IP | Transmission Control Protocol / Internet Protocol |
| TF*IDF | Term Frequency Inverse Document Frequency |
| TXT | Text |
| UDP | User Datagram Protocol |

| | |
|---|---|
| UMDL | University of Michigan Digital Library |
| UML | Unified Modeling Language |
| URL | Uniform Resource Locator |
| WAIS | Wide Area Information Server |
| WWW | World Wide Web |

# CHAPITRE 1
# INTRODUCTION

De nos jours, l'Internet, et tout particulièrement la toile mondiale du Web, prennent une place d'avant scène dans le quotidien ; Ils sont et est devenu un support informationnel privilégié parmi les médias. Alors que le téléphone a dû attendre 48 ans depuis sa découverte pour rejoindre 50 millions d'abonnés, la télévision a mis 13 ans, le câble 10 ans et l'Internet seulement 5 ans (Murch et Johnson, 1998). De par son accessibilité, il est rapidement entré dans notre quotidien comme espace de communication, de commerce, de divertissement, et est devenu une source principale d'information électronique. De ce fait, l'information a connu une croissance exceptionnelle de sa diffusion sous forme numérique sur le Web. Cette croissance phénoménale, puisqu'elle entraîne une répartition et une dissémination de l'information, a lancé le grand défi de la recherche des données pertinentes dans ce vaste océan d'informations. Force est de constater qu'il est assez difficile pour un utilisateur non chevronné du Web de trouver, dans un temps raisonnable et sans efforts faramineux de tri et de filtrage, les informations qui l'intéressent dans cet immense entrepôt mondialisé de données électroniques qui est le Web. Ainsi, il devient nécessaire de mettre à la disposition de l'utilisateur, des mécanismes ou des outils informatiques adaptés pour assister sa recherche d'information. La mise en place de tels outils constitue l'objet du présent mémoire. Dans ce premier chapitre, les limitations des outils de recherche existants et les problèmes inhérents à l'arrivée massive des divers technologies et supports de l'information sont abordés. Nous présentons également nos objectifs de recherche, les résultats escomptés et la contribution de nos travaux.

## 1.1 Motivations et le contexte de recherche

À des fins de recherche de données informatisées sur Internet ou sur Intranet, l'utilisateur a principalement à sa disposition des moteurs ou engins de recherche. Dans

cette section, nous décrirons d'abord les caractéristiques et les principales fonctionnalités offertes par ces outils, puis nous esquisserons les problèmes et les limitations rencontrés lors de leur utilisation. Nous présenterons enfin certains outils plus complets, à savoir les bibliothèques virtuelles qui apportent de nouvelles fonctionnalités à la recherche d'information.

### 1.1.1 Moteurs ou engins de recherche sur Internet

Un moteur de recherche, appelé aussi engin de recherche, est un outil informatique mis, bien souvent gratuitement, à la disposition des utilisateurs pour qu'ils puissent effectuer leurs recherches d'information sur Internet. Ces outils suivent quatre logiques de recherche : la recherche géographique, la recherche thématique, la recherche par index et la recherche par méta-index (Samier et Sandoval, 1998).

***Recherche géographique :*** Les outils de recherche géographiques permettent de trouver par localisation géographique un ou plusieurs sites Web. Les sites proposés par ces outils sont répertoriés par ville. L'utilisateur de ces engins a accès à des cartes géographiques, des listes de pays ou de villes. Par simple sélection, les zones géographiques se précisent et leur niveau de détail s'élargit pour aboutir finalement à la liste des sites hébergés dans une ville. Parmi les nombreux outils de recherche géographique, citons à titre d'exemple www.excite.com/travel/ et www.urec.cnrs.fr/annuaire/ .

***Recherche thématique :*** Les outils de recherche thématique correspondent à des catalogues thématiques qu'on peut consulter à l'aide d'une liste de choix de mots-clés (un mot clé correspond à un thème). Les thèmes, ainsi que le choix de répertorier une adresse de site, sont préétablis pour chacun des outils. De thème en sous-thème et étape par étape, ces outils fournissent une liste d'adresses de sites correspondant aux mots-clés choisis. Parmi les outils de recherche thématique, nous avons retenu les suivants, qui sont par ailleurs les principaux : www.yahoo.com (section thématique), magellan.excite.com, www.einet.net, www.nomade.fr.

***Recherche par index :*** Les outils de recherche par index sont les moteurs de recherche privilégiés des internautes, au point qu'ils sont considérés souvent comme les seuls moteurs de recherche. Nous allons décrire dans ce qui suit le principe de base de la recherche par index ; celle-ci sera présentée d'une manière plus formelle au chapitre suivant.

Le processus de recherche se déroule de la façon suivante : à partir d'une requête formulée par l'utilisateur, à l'aide de mots-clés et éventuellement d'opérateurs booléens (ET, OU, NON, etc.), auprès d'un moteur de recherche, ce dernier répond en donnant généralement le nombre de documents contenant les mots exprimés dans la requête, le titre de chacun des documents, leur adresse Internet, leur résumé et le pourcentage de pertinence de la réponse. À partir de l'analyse et du filtrage manuels de ces résultats, l'utilisateur explorera chacun des sites ainsi trouvés.

Les moteurs de recherche fonctionnent généralement en trois étapes :

1. Un "robot informatique" (outil informatique automatisé) visite les sites Internet et recopie les documents dans un entrepôt de données temporaire situé dans le site d'administration du moteur de recherche. Cette étape correspond à une collecte d'information sur les Web, Gopher, Wais, News ou Ftp. Chaque moteur de recherche détermine librement le type de source qu'il veut scruter.
2. Un algorithme d'indexation classe les documents de l'entrepôt de données temporaire. Ceci se fait par une indexation systématique de tous les mots provenant de tous les documents. L'index ainsi généré représente désormais une image de l'ensemble des documents collectés.
3. Lorsqu'une requête est formulée, le moteur de recherche parcourt l'index en déterminant des ensembles de documents contenant chacun un des mots clés de la requête. Finalement, le moteur applique sur ces ensembles les opérations ensemblistes correspondant chacune à un opérateur booléen ou logique spécifié par l'utilisateur dans sa requête.

***Recherche par méta-index :*** Les outils de recherche par méta-index, dits méta-moteurs, se situent au-dessus des moteurs de recherche. Ils remplissent la fonction de rechercher en parallèle dans plusieurs moteurs de recherche, et de fournir rapidement les résultats compilés. Ces moteurs de recherche ne sont pas des outils "par index" puisqu'ils n'ont pas d'index propre sur leur site. Ce sont des programmes informatiques qui réalisent une interface évoluée et rapide entre les moteurs et l'internaute. À partir d'une question posée sur son site, un méta-moteur traduit la requête et la soumet simultanément à plusieurs moteurs de recherche qui fournissent les réponses. Il reçoit les 10 à 50 premières réponses de chaque moteur, puis les compile en éliminant les réponses en double et fournit la localisation des documents (titre, adresse et résumé), ainsi que le nom des moteurs qui ont fourni les résultats. Parmi les principaux méta-moteurs, citons : www.metacrawler.com, www.copernic.com, www.profusion.com, www.askjeeves.com.

### 1.1.2 Problèmes et limitations des moteurs de recherche

Bien que les moteurs de recherche aient subi, depuis leur création, des améliorations de leur puissance et de leur capacité de traitement de gros volumes de données, ils continuent aujourd'hui à présenter à leurs utilisateurs des problèmes de taille qui rendent leur utilisation presque inefficace et insatisfaisante.

L'un des handicaps majeurs des moteurs de recherche est la nature anarchique intrinsèque à Internet. En effet, en raison de son développement rapide et désordonné, Internet est devenu un réseau d'informations sans organisation ni structure. Selon le cas, les informations sont justes, fausses, précises, floues ; la propagande avoisine les documents officiels et les inepties. Ainsi, toute recherche (non affinée) par index, comme celle effectuée par les moteurs de recherche, mène à une identification d'un très grand volume de réponses, dont la plupart sont non pertinentes. L'utilisateur face à un tel volume se verra vite découragé et épuisé par un tri manuel. À la requête formulée par le mot-clé "avion", le moteur retournera 110 000 pages ; avec le mot-clé "*plane*", on obtient près d'un million de pages (requêtes effectuées sur le moteur de recherche *Altavista*).

Comment localiser rapidement l'information voulue, et seulement elle, tout en étant sûr qu'elle soit authentique et fiable ?

Un deuxième point faible des moteurs de recherche est l'ordre de classement des résultats. Il est très difficile de mesurer la pertinence d'une réponse à partir de mots-clés, celle-ci dépendant du profil de l'utilisateur. Les moteurs de recherche utilisent habituellement des règles empiriques pour ordonner les résultats. Ces règles, appelées fonctions de classement, sont appliquées aux requêtes affinées (plusieurs mots-clés et opérateurs booléens) et aux requêtes générales (un mot-clé).

Un exemple répandu de fonction de classement est une fonction qui prend la valeur du nombre de fois où le mot-clé apparaît dans une page donnée. Le classement des résultats correspond à l'ordre décroissant des valeurs de cette fonction. Un autre exemple de fonction de classement considère la position relative du mot-clé dans la page. On place en premier la page dont le mot-clé apparaît le plus tôt. Ces approches ne donnent pas toujours de bons résultats (Charkrabarti et al., 1999). En effet, de nombreux sites abusent de ces règles, afin que leurs pages soient bien classées. C'est pour cette raison que le titre de certaines pages est étrangement libellé tel que "billets avion soldés, billets avion soldés, billets avion soldés", avec une redondance artificielle de certains mots-clés. D'autres sites incluent plusieurs fois dans leurs pages des phrases judicieusement choisies, dans des polices et des couleurs invisibles à l'œil. De telles pratiques qui inondent les moteurs de recherche avec des informations non pertinentes (connues dans le jargon de l'Internet sous le nom de "*spamming*"), compliquent la tâche des moteurs.

Le troisième inconvénient des moteurs de recherche provient de l'abondance de synonymes et de mots polysémiques (des mots qui ont plusieurs sens). Une demande d'information sur les "automobiles" laissera de côté une kyrielle de pages qui évoquent les "voitures". En raison des polysémies, une simple requête sur "Jaguar" donnera des milliers de pages concernant, entre autres, l'automobile de ce nom, le félin et une équipe de football américain. On améliore la recherche en utilisant les relations sémantiques entre les mots. Certains moteurs de recherche utilisent de telles relations généralement

identifiées par les linguistes et nommées "réseaux sémantiques". À la réception de la requête "automobile", un moteur de recherche couplé à un réseau sémantique détermine d'abord que le mot "automobile" est équivalent au mot "voiture", puis il donne la liste des pages qui contiennent l'un des deux mots. Toutefois, cette méthode est à double tranchant : elle résout le problème des synonymes mais aggrave celui de la polysémie.

### 1.1.3 Bibliothèques virtuelles

Bien que ne remédiant pas à tous problèmes cités ci-dessus, le concept des bibliothèques virtuelles a été élaboré afin de combler les besoins d'un utilisateur en enrichissant le noyau d'un moteur de recherche classique avec une panoplie de services semblables à ceux qu'un utilisateur peut recevoir dans une bibliothèque traditionnelle. D'une manière formelle, une bibliothèque virtuelle, appelée aussi dans la littérature scientifique bibliothèque numérique (*Digital Library*), est un système informatique donnant accès à une interface personne-machine conviviale et homogène permettant d'interroger et de consulter un grand volume de données réparties (dispersées géographiquement) et numérisées sous différents formats, et offrant les services d'un bibliothécaire (Paepcke et al., 1998). Bien que la recherche scientifique dans le domaine des bibliothèques virtuelles date de plus de 10 ans, il n'existe toutefois pas encore un standard décrivant les services et les fonctions essentielles à ces systèmes. Nous pouvons néanmoins donner les caractéristiques les plus répandues :

- Une des caractéristiques des bibliothèques virtuelles est de faciliter les moyens de recueil, de stockage et d'organisation de l'information numérique. Aussi rendent-elles facilement accessible, exploitable et traitable l'information recherchée depuis un réseau de communications.
- Alors que les moteurs de recherche puisent leurs informations numériques principalement de l'Internet, les bibliothèques virtuelles ne se donnent pas cette limitation et ouvrent la voie, du moins théoriquement, à la recherche d'informations numériques provenant de sources d'informations hétérogènes.

Autrement dit, les sources d'informations impliquées dans une bibliothèque virtuelle peuvent être autant des sites Internet, que des bases de données spécialisées, ou tout autre source d'informations numérisées.

- Contrairement aux moteurs de recherche qui sont gratuitement mis à la disposition des utilisateurs, les bibliothèques virtuelles sont souvent privées : elles appartiennent à des organismes universitaires ou industriels et donnent accès aux seuls utilisateurs abonnés.

Bien que le domaine des bibliothèques virtuelles concentrent l'essentiel des efforts de recherche dans le domaine de la recherche d'information numérisée, toutefois les bibliothèques virtuelles sont, pour la plupart, encore à l'état de projets ou de prototypes. Notons ici quelques cas d'exceptions, telles que les bibliothèque virtuelles des organismes de publications scientifiques comme ACM (*Association for Computing Machinery*) ou IEEE (*Institute of Electrical and Electronics Engineers*), mais dont la panoplie de services est limitée et dont la recherche se limite uniquement à leurs propres bases de données informationnelles.

Parmi les projets de bibliothèques virtuelles développés à date, DLI (*Digital Library Initiative*) est le plus ambitieux des projets. Il est subventionné, entre autres, par la NSF (*National Science Foundation*), la DARPA (*Department of Defense Advanced Research Projects Agency*), et la NASA (*National Aeronautics and Space Administration*). Ce programme de recherche est issu du regroupement de six projets multidisciplinaires. Chacun de ces projets rassemble des équipes de chercheurs provenant d'une université pilote en collaboration avec d'autres organisations. Plus de 75 organisations distinctes ont créé des relations associatives avec ces projets. Les organisations associées représentent des intérêts variés et comprennent de grandes compagnies américaines d'informatique et de communications, des institutions académiques de tous niveaux, des bibliothèques, des éditeurs, des agences gouvernementales et étatiques, des associations professionnelles, et d'autres organisations engagées dans les bases de données et la gestion de l'information à grande échelle (Griffin, 2000).

Les six grandes universités américaines associées au projet DLI ont chacune un domaine de travail spécifique. L'Université de Californie à Berkeley travaille spécifiquement à la planification des environnements et des systèmes d'information multimédiatisés, elle a mis au point le projet Alexandria concernant l'information géographique. L'université Carnegie Mellon s'intéresse aux bibliothèques d'infomédia numérique vidéo. L'Université de l'Illinois concentre ses travaux sur le dépôt fédéré de littérature scientifique; l'Université du Michigan s'occupe des agents intelligents pour la localisation de l'information et l'université Stanford mène des travaux sur les mécanismes interopérationels permettant l'offre de services hétérogènes.

Parmi ces diverses équipes de recherche, celle de l'Université du Michigan s'est distinguée par son travail de pionniers dans la recherche et la localisation de l'information assistée par des outils informatiques intelligents. Leur projet UMDL (*University of Michigan Digital Library*) est une architecture multi-agents basée essentiellement sur la notion d'agents de recherche d'informations numérisées. Cette architecture phare est devenue un modèle de référence pour divers travaux de recherche dans le domaine. Les agents de recherche d'informations numérisées et les agents intelligents sont au cœur du présent mémoire et feront l'objet d'une attention particulière au cours du chapitre 2.

## 1.2 Éléments de la problématique

Le principal défi posé à la conception d'un outil de recherche d'information numérisée est celui de l'efficacité. En effet, la raison d'être de l'outil est de résoudre, en premier lieu, le problème de la pertinence des résultats parmi le volumineux ensemble de réponses que l'on peut obtenir. Ainsi, découlent les questions suivantes auxquelles un tel outil devrait répondre : Quel sera le langage de requête mis à la disposition des utilisateurs ? Quel type de recherche effectuera-t-il ? Quels algorithmes ou heuristiques seront-ils utilisés lors des recherches ? Comment le filtrage des résultats se fera-t-il ? Quel sera alors le classement d'un document en fonction de sa pertinence face à la requête client ?

Comment l'outil affinera-t-il son degré d'efficacité face à la connaissance qu'il pourra développer d'un profil client ? De quelle manière l'outil pourra apprendre de ses expériences ? Quelle sera la marge de manœuvre (autonomie) et le degré de confiance alloués à l'outil ? Autrement dit, quel sera le degré d'intelligence d'un tel outil ?

D'autre part, la diversité et l'hétérogénéité des formats et des sources de données numérisées posent à la conception de l'outil un problème de taille. Quel sera le format de données supporté par la recherche d'un tel outil (texte, audio, images, vidéo) ? Quel type de sources d'informations numérisées un tel outil supportera-t-il (Internet, bases de données, bases de connaissances, etc.) ? Peut-on offrir à l'aide d'un seul outil une interface homogène ou universelle pour interroger tout type de sources d'informations ?

Enfin, l'aspect dynamique de l'information numérisée disponible sur l'Internet rend la conception de l'outil plus complexe. En effet, les informations provenant par exemple du Web sont assujetties à des mises à jour fréquentes et presque quotidiennes. Ainsi, un outil de recherche d'information efficace devra prendre en compte ce dynamisme et comporter des mécanismes de mises à jour régulières de ses sources d'informations. La question qui se pose alors est de savoir quels seront les mécanismes qui permettront à cet outil une adaptation rapide face au dynamisme et à la prolifération des sources d'informations numérisées disponibles.

Face à ces défis de taille, nous allons nous restreindre dans notre recherche essentiellement à l'étude de certains de ces problèmes que nous préciserons dans la prochaine section.

## 1.3 Objectifs de recherche et les résultats escomptés

L'objectif principal de ce mémoire est de concevoir un modèle d'agent de recherche intelligente d'information numérisée. Le but essentiel de cet agent est d'améliorer la pertinence des résultats d'une recherche en intégrant la notion de profil de l'utilisateur. Nous nous limiterons dans ce mémoire à un seul type de recherche d'information numérisée, à savoir la recherche en mode texte excluant ainsi toute information

numérisée sous forme image, audio ou vidéo. Nous nous limiterons aussi à un seul support d'information, à savoir Internet.

Il s'agit donc de définir un modèle permettant de mettre en place des agents d'information qui répondent aux critères suivants :

- limiter le volume des réponses aux requêtes de l'utilisateur, en éliminant et en filtrant celles qui ne sont pas pertinentes en fonction d'un profil préétabli de l'utilisateur en question ;
- mettre en place un système d'évaluation du degré de satisfaction de l'utilisateur afin de permettre à l'agent d'affiner ses réponses futures en fonction de cette évaluation ;
- intégrer les mécanismes nécessaires à la mise à jour fréquente des connaissances de l'agent face au dynamisme des sources d'informations.

Nous désirons aussi fournir à l'intérieur de ce modèle tous les éléments et mécanismes de base de la recherche intelligente d'informations, ainsi que les algorithmes essentiels à une telle activité.

## 1.4 Contribution et originalité des travaux

Le modèle d'agent présenté dans le cadre de ce mémoire s'inscrit essentiellement dans le cadre de la recherche intelligente d'information sur Internet. Ce modèle tente de remédier à certaines faiblesses inhérentes aux moteurs de recherche par index et de tirer profit des nombreux avantages présentés par la recherche d'information basée sur l'espace vectoriel tels que mentionnés à la section 2.2. Ainsi, le modèle proposé dans le chapitre 3 est largement inspiré de l'ensemble des travaux de recherche présentés au chapitre 2, plus particulièrement de la notion d'espace vectoriel de recherche d'information introduite par Salton et McGill (1983).

Le modèle d'agent proposé apporte plusieurs contributions aux notions de profil d'utilisateur, de formulation de requête, de raffinement des requêtes et de filtrage et présentation des résultats, propres à la théorie de recherche d'information. Au niveau

pratique, notre modèle propose l'utilisation du profil de l'utilisateur afin de soulager une formulation complexe des requêtes et de cibler les résultats pertinents aux besoins informationnels de l'utilisateur en question. Sur le plan de la viabilité, l'implantation partielle de notre modèle, telle que décrite au chapitre 4, a donné lieu à un prototype qui a montré, lors d'une expérimentation comparative avec l'engin de recherche par index *Altavista*, une grande fiabilité et une pertinence améliorée dans le classement des résultats. Finalement, puisqu'il est essentiellement lié au problème de recherche d'information, le modèle proposé contribue à ce domaine par la mise en place d'éléments permettant aux utilisateurs de définir leurs besoins et aux responsables du système de recherche d'information d'encadrer et de définir le contenu et l'étendue de l'information par l'entremise d'un mécanisme de collecte d'informations à partir d'une liste embryonnaire de documents.

## 1.5 Organisation du mémoire

Le prochain chapitre propose une revue sélective de littérature sur les méthodes et outils de recherche d'informations numériques, les agents d'information et les méthodologies servant de fondements aux travaux de ce mémoire. Le chapitre 3 décrit le modèle conceptuel de l'agent de recherche intelligente d'informations numérisées que nous proposons. Le chapitre 4, quant à lui, précise les choix d'implantation d'un tel agent et détaille les résultats d'expérimentation. Finalement, le chapitre 5, en guise de conclusion, présente une synthèse des travaux et des principales contributions de ce mémoire, pour finalement en indiquer les limitations et préciser les orientations de recherches futures.

# CHAPITRE 2
# OUTILS ET MÉTHODES DE RECHERCHE D'INFORMATION

Comme nous l'avons vu dans le chapitre précédent, les moteurs de recherche et, en particulier, ceux basés sur une recherche par index, ainsi que les bibliothèques virtuelles et, en particulier, celles basées sur les systèmes multi-agents, sont les principaux outils mis à la disposition des utilisateurs dans le cadre de la recherche d'informations numérisées. Dans ce chapitre, nous présentons les principes de base de la recherche d'information et les méthodes employées généralement par les moteurs de recherche par index. Par la suite, nous décrivons les bibliothèques virtuelles ainsi que les agents et les systèmes multi-agents sur lesquels reposent la plupart des architectures. Finalement, nous définissons ce qu'est la recherche intelligente d'information dans le contexte de ce mémoire et nous présentons quelques approches et modèles sous-jacents.

## 2.1 Moteurs de recherche par index

Bien que l'information numérisée puisse être disponible sous différents formats, tels que HTML, TXT, PDF, etc., et dans différentes sources sur différents supports tels que les bases de données, les bases de connaissances, le WEB, les *News Group*, etc., les moteurs de recherche d'information s'intéressent uniquement aux documents ayant des formats et des supports qui permettent une recherche en mode texte. Autrement dit, l'élément et l'unité atomique de base manipulés par ces moteurs sont les mots provenant de l'ensemble des documents recherchés. De plus, dans leur majorité, les moteurs de recherche tiennent compte uniquement de l'information textuelle numérisée sur Internet et particulièrement celle présente sur le Web. La Figure 2.1 illustre le fonctionnement général des moteurs de recherche.

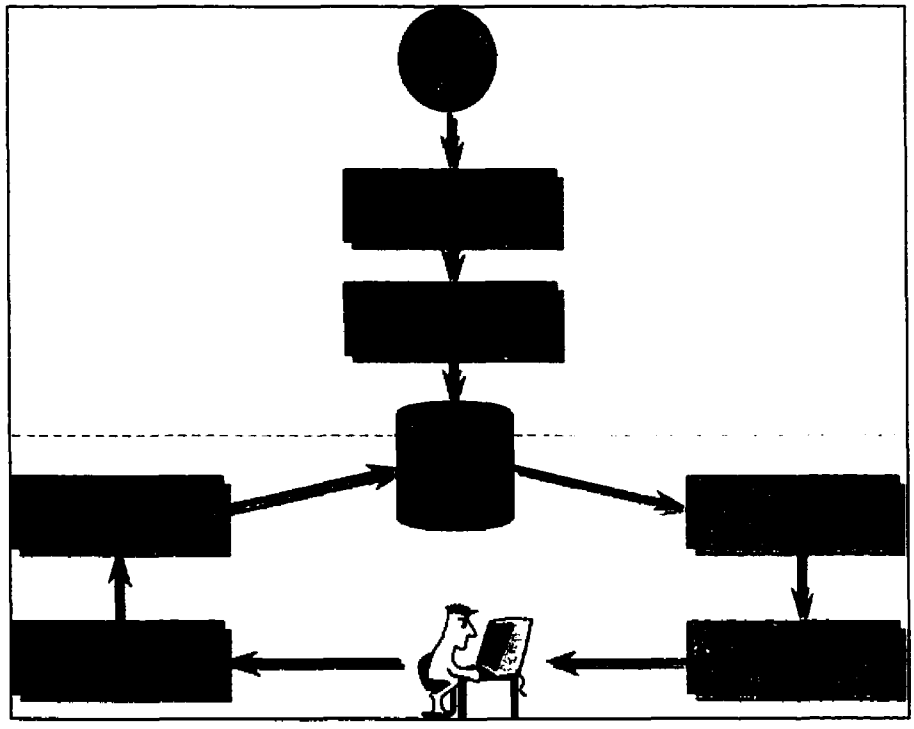

**Figure 2.1 Fonctionnement général des moteurs de recherche**

Un moteur de recherche se caractérise par le temps de réponse en fonction du volume de l'index parcouru (la puissance de calcul), la richesse du langage de requête ou des opérateurs logiques disponibles, le modèle de fonctionnement correspondant à l'algorithme d'indexation des documents utilisé, et la diversité des sources (FTP, Gopher, WAIS, Web, News Group) (Samier et Sandoval, 1998). Dans ce qui suit, nous allons décrire les différentes étapes du cycle de fonctionnement d'un moteur de recherche.

### 2.1.1 Parcours des pages

Généralement, l'activité de recherche effectuée par les moteurs est basée sur une indexation préalable de tous les documents disponibles. Toutefois, avant d'indexer l'ensemble des documents présents sur le Web, il faut les parcourir afin de collecter l'information s'y trouvant. Un tel parcours, pour qu'il soit exhaustif, nécessite une méthodologie systématique. Or, le Web est un graphe orienté, dont les nœuds sont les

documents HTML et les arcs sont les liens hypertextes entre ces documents (Caglayan et Harrison, 1998). Ce graphe est noté : $G = (N, A)$, où $N$ désigne l'ensemble des nœuds représentant respectivement les documents ou pages logés sur le Web et $A$ l'ensemble des arcs représentant respectivement les hyperliens entre ces documents.

Ainsi, le problème de parcourir le Web, se ramène simplement à celui de parcourir le graphe $G$. Afin d'éviter les cycles, un parcours exhaustif de $G$ revient à parcourir les $|N|$ documents ou nœuds d'un arbre couvrant le graphe $G$ (problème classique de recherche opérationnelle connu sous le nom du *Spanning Tree*).

En raison de la nature dynamique des documents sur le Web, du nombre impressionnant et perpétuellement croissant de ceux-ci, et par conséquent du nombre de nœuds du graphe que l'on veut parcourir, en plus de la possibilité d'avoir plus qu'une seule composante connexe dans un tel graphe, les moteurs de recherche utilisent souvent des stratégies de parcours partiel et appliquent des méthodes heuristiques lors de ce parcours partiel. Par exemple, la stratégie de priorité adoptée par le moteur de recherche *Lycos* est la suivante : Quand une page Web est extraite, le moteur conserve tous les hyperliens de cette page dans une file d'attente. Par hyperliens, nous entendons les adresses URL indiquant la localisation des pages du Web sous une forme standard telle que http://www.polymtl.ca . Le moteur choisit ensuite la prochaine page à ramener, à partir de la file d'attente, avec une stratégie aléatoire qui favorise les pages qui ont été référencées plus d'une fois par d'autres pages, ainsi que celles qui possèdent une adresse URL parmi les plus courtes, pour pouvoir explorer les répertoires parents en premier (par exemple le répertoire http://www.polymtl.ca/etudes/ est un répertoire parent du répertoire http://www.polymtl.ca/etudes/bacc/). La plupart des autres moteurs de recherche utilisent une stratégie plus simple, qui consiste à parcourir le graphe $G$ en adoptant une stratégie connue sous l'appellation de parcours en largeur.

Comme illustré à la Figure 2.2, le parcours et la collecte sont effectués par une composante dédiée de l'architecture générale d'un moteur de recherche, appelée le robot du Web (*Web robot)* (Etzioni et Weld, 1995).

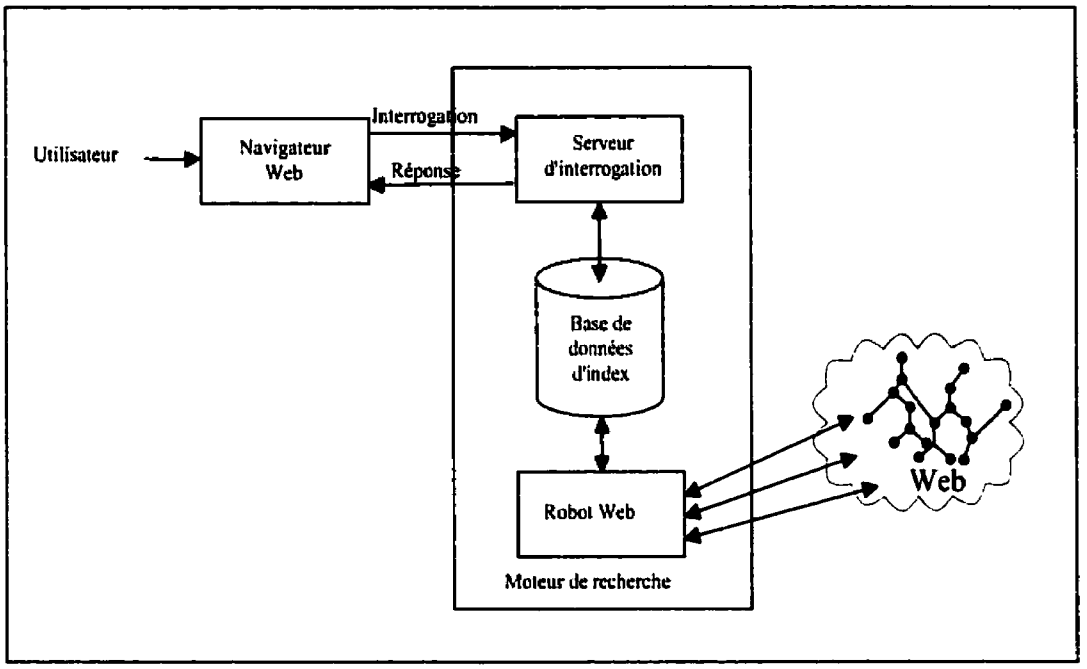


**Figure 2.2 Architecture générale d'un moteur de recherche**

La collecte des documents, première étape nécessaire à la recherche d'information sur Internet, pose encore de nombreux problèmes. L'étape de collecte des documents est considérée, par beaucoup, comme triviale, et déjà suffisamment efficace pour être pleinement opérationnelle. Cependant, comme nous venons de le voir, il ne faut pas s'y tromper, de nombreux efforts sont encore à fournir. Comme le dit Louis Monier (Sullivan, 1999), créateur d'*AltaVista*, un document a une réelle existence sur Internet si, et seulement si, il est référencé par un moteur de recherche. En effet, d'après de récentes statistiques, 1 document sur 28 visionné sur le Web correspond à un résultat d'un moteur de recherche (3.5 % de toutes les pages visionnées) (Alexa, 2000). Alors, pourquoi tous ces moteurs de recherche occultent encore un si grand nombre de documents : les documents n'utilisant pas l'alphabet latin, les documents dynamiques et les documents stockés dans des formats propriétaires, pour n'en citer que quelques uns ?

### 2.1.2 Indexation

L'indexation des documents collectés est une étape très importante dans le processus de recherche. En effet, la qualité de la recherche dépend de la qualité de l'indexation. L'indexation des documents collectés devrait permettre d'extraire l'information pertinente de ces derniers. Cette étape devrait nécessiter, pour être réellement efficace, une extraction intelligente des structures du document (mots, phrases, sigles) ; une analyse morphologique, une analyse syntaxique, une analyse grammaticale et la lemmatisation[1] des mots ; l'élimination des mots vides et l'extraction des mots "pertinents" d'un document.

Nous décrivons ici des processus et traitements nécessitant une puissance de calcul démesurée pour être appliquée à l'ensemble du Web. Les moteurs actuels, comme nous le verrons, sont très loin d'approcher de telles possibilités ; ils privilégient une indexation rapide et donc très simple. C'est la raison pour laquelle nous n'avons retenu que quatre critères essentiels: l'extraction des mots du document, la normalisation des mots, l'élimination des mots vides et l'indexation.

***L'extraction des mots du document***

C'est une étape du processus d'indexation qui peut sembler triviale au premier abord, et qui pourtant constituera la base de tout le reste du processus d'indexation. Il faut donc que cette phase soit d'une qualité maximale. L'approche de l'ensemble des développeurs des moteurs de recherche est très simple : ils considèrent un mot comme étant une chaîne de caractères correspondant à l'expression régulière : **[A-Za-z\-\'$accents]+**

où **$accents** est une liste de tous les accents possibles dans la table ISO-8859-1 (ALIS, 1996).

---

[1] Chaque mot est normalisé : *belle, belles, beau, beaux* seront normalisés en *beau*; *marcher, marchèrent, marcheras* seront normalisés en *marcher*, etc.

Cette expression régulière signifie qu'un mot est défini comme une chaîne constituée d'au moins un caractère (+) et pouvant contenir :

- n'importe lesquelles des 26 lettres de l'alphabet en majuscule (A-Z),
- n'importe lesquelles des 26 lettres de l'alphabet en minuscule (a-z),
- le tiret (\-),
- l'apostrophe (\'),
- n'importe lesquels des caractères accentués du jeu de caractère : **$accents**.

Cette définition (simpliste) a été retenue par *AltaVista, InfoSeek Ultra, Excite, AliWeb, OpenText* et *Carrefour.net.*

***La normalisation des mots (lemmatisation)***

Ce traitement consiste à retrouver pour un mot sa forme normalisée (généralement le masculin pour les noms, l'infinitif pour les verbes, le masculin-singulier pour les adjectifs, etc.). Ainsi, dans l'index ne sont conservées que les formes normalisées, ce qui offre un gain de place appréciable, mais surtout, si le même traitement est effectué sur la question, cela permet une plus grande souplesse et rapidité de la recherche.

***L'élimination des mots vides (listes d'arrêt ou "Stop List")***

Ce processus est également important dans la mesure où c'est un facteur qui a une grande influence dans la précision de la recherche. En effet, le fait de ne pas éliminer les mots vides provoque inévitablement des réponses non pertinentes. L'élimination des mots vides (le, la, et, etc.) doit se faire aussi bien lors de l'indexation que lors de la recherche (élimination des mots vides de la question).

Certains moteurs (*AltaVista, Excite, Lycos*) suppriment les mots vides en utilisant la fréquence d'apparition des mots dans l'ensemble des documents de la base. Si un mot apparaît trop couramment, il est considéré comme trop fréquent, donc non-informationnel, et il ne sera donc pas pris en compte lors de la recherche. Ce système peut aboutir à des aberrations. Les plus flagrantes sont visibles avec *AltaVista*. Par

exemple, pour la requête "information retrieval system on the Internet", les termes "information", "system", "on", "the" et "Internet" apparaissent trop fréquemment dans la base et sont donc ignorés. La recherche ne s'effectue donc que sur le terme "retrieval".

D'autres moteurs (*EuroFerret, Galaxy, InfoSeek, Magellan, WebCrawler, WWWWorm*) utilisent une liste des mots vides (une sorte de dictionnaire). Enfin, des moteurs comme *AliWeb, HotBot* ou *OpenText* ne se soucient guère des mots vides, pour eux, tous les mots sont informationnels.

***L'indexation***

L'indexation proprement dite consiste à retenir les termes les plus significatifs du document. Il existe plusieurs approches : certains moteurs se contentent d'indexer uniquement les mots des titres (en se basant sur des balises HTML) du document, certains indexent systématiquement tous les mots du document. Parmi les derniers, *Lycos* est l'un des seuls à retenir un certain nombre de mots significatifs (il en retient une vingtaine). Mais, nous n'avons actuellement aucune information sur les critères lui permettant de retenir tel ou tel mot.

Comme nous venons de le voir, l'indexation réalisée par les moteurs de recherche actuels reste très rudimentaire, mais en revanche très rapide. Cette rapidité de l'indexation permet d'obtenir une fréquence de mise à jour élevée de l'index pour une couverture très large des ressources disponibles sur Internet. Cependant, la qualité de l'indexation est souvent médiocre, ce qui conduit généralement à une grande quantité de réponses non pertinentes. Si l'exhaustivité des documents indexés sur Internet permet de fournir un certain nombre de documents pertinents, l'utilisateur est souvent noyé dans la quantité des résultats.

### 2.1.3 Interrogation

Dans le contexte des moteurs de recherche, il n'existe pas un langage de requête universel, ni de standard communément adopté. Néanmoins, nous pouvons distinguer différents modes de formulation d'une requête. Trois types d'interrogation sont

envisageables : l'interrogation booléenne, l'interrogation par liste de mots et l'interrogation en langage naturel.

***L'interrogation booléenne***

L'interrogation booléenne est le type d'interrogation le plus rudimentaire, et surtout le plus simple à mettre en place. Il consiste à formuler une question avec une liste de termes séparés par des opérateurs logiques (AND, OR, NOT), et à rechercher les documents correspondant à cette requête. La Figure 2.3 illustre d'une manière formelle mais simplifiée la grammaire du langage de requête usuellement utilisé pour formuler des interrogations booléennes. Par exemple, pour la question "(moteur AND recherche) OR (search AND engine)" (les systèmes actuels n'étant pas multilingues, c'est le seul moyen d'effectuer une recherche à la fois en français et en anglais!), le système doit fournir comme réponse tous les documents de la base contenant les deux termes "moteur" et "recherche" ou "search" et "engine".

**<lettre>** → a | b | ...| z | A | B | ... | Z

**<chiffre>** → 0 | 1 | 2 | ... | 9

**<caractère spécial>** → _ | - | ' | \ | . | !

**<mot>** → **<lettre>** | **<chiffre>** | **<caractère spécial>** { **<mot>**}

**<opérateur booléen>** → AND | OR | NOT

**<requête>** → **<mot>** { **<mot>** | **<requête>** } | **<mot>** **<opérateur booléen>** **<mot>**

**Figure 2.3 Grammaire simplifiée du langage de l'interrogation booléenne**

À l'heure actuelle, un grand nombre de systèmes disponibles sur Internet fonctionnent suivant ce type de recherche (*AltaVista, Excite, Galaxy, Harvest, HotBot, InfoSeek, Lycos, Magellan, OpenText, WebCrawler, WWWWorm*). Notons que la syntaxe "+/-" utilisée par certains moteurs de recherche est tout à fait similaire à celle des opérateurs

logiques. En effet, ces systèmes considèrent les coupures de mots (les espaces entre les mots) comme des opérateurs OR implicites, l'opérateur "+" revient à un AND, puisqu'il permet d'imposer la présence d'un mot, et l'opérateur "-" permet de proscrire la présence d'un mot (équivalent au NOT). Parmi les raffinements fournis par certains systèmes afin de pallier les limitations de la recherche booléenne, mentionnons : la combinaison des opérateurs, la mise en parenthèses, la troncature et les opérateurs de proximité et d'adjacence.

La ***combinaison des opérateurs*** (comme dans notre exemple) va permettre d'effectuer des recherches un peu plus complexes que celles proposées par *OpenText, WebCrawler* ou *WWWWorm* qui ne permettent d'utiliser qu'un seul opérateur entre les différents mots. Ainsi, on peut rechercher "Moteur AND recherche AND search AND engine", ou "Moteur OR recherche OR search OR engine", mais vous n'avez aucun moyen d'effectuer une recherche avec une question du type "(Moteur AND recherche) OR (search AND engine)". C'est une limitation très contraignante quand on veut effectuer une recherche booléenne efficace.

La ***mise en parenthèse*** des expressions permet de compliquer un peu plus les requêtes, en permettant à l'utilisateur averti d'effectuer des recherches complexes. Cette possibilité, qui pourtant semble simpliste et tout à fait naturelle dans un contexte booléen, n'est cependant disponible que dans un nombre limités de moteurs de recherche *(AltaVista, Excite).*

La troncature (automatique et/ou manuelle) permet de rechercher des sous-chaînes. L'opérateur de troncature (généralement "*") remplace un ensemble de caractères afin d'effectuer des recherches plus larges. On distingue trois types de troncature : la troncature droite (la plus commune), la troncature gauche, et enfin la troncature interne. La troncature droite permet d'effectuer une recherche en utilisant le début d'un mot, afin d'obtenir les différentes formes dérivées du mot (et d'autres mots n'ayant aucun rapport

avec ce que recherche l'utilisateur!). Par exemple, si vous recherchez des documents sur les chats, vous pouvez saisir la requête "chat*" afin d'obtenir les documents contenant les termes chat, chats, chatte, chattes, chaton, chatons, ... Le résultat va cependant être surprenant, puisque les documents contenant les termes français chatoiement, chatouille, chatterton seront également retournés. De plus, puisque le Web contient essentiellement des documents de langue anglaise, vous obtiendrez de nombreux documents contenant les mots *chat* (bavardage), ou *chattel* (bien mobilier), qui n'ont rien à voir avec la recherche. Les troncatures gauche et interne fonctionnent de la même manière, elles permettent respectivement de rechercher des chaînes sans spécifier le début du mot ou une partie interne du mot.

Tous ces opérateurs sont à manipuler avec précaution. En effet, en lançant une recherche avec troncature dans un environnement multilingue comme le Web, la non pertinence devient rapidement trop importante pour pouvoir exploiter les résultats obtenus.

Les ***opérateurs de proximité et d'adjacence*** ont pour but de contraindre le système à rechercher des mots se trouvant proches l'un de l'autre, c'est-à-dire ayant des liens syntaxiques et/ou sémantiques. Ainsi, en posant la question "moteur ADJ recherche", on élimine un certain nombre de documents concernant la recherche dans le domaine des moteurs automobiles ou aéronautiques. Cependant, cet opérateur ne permet pas de spécifier qu'il doit exister un lien syntaxique entre les mots. Il va considérer qu'un document est pertinent si les mots "moteur" et "recherche" ne se trouvent pas séparés dans le texte par plus de 2, 3, 4, 5, ...,10 mots. La limite du nombre de mots est, soit déterminée arbitrairement par le système, soit à spécifier par l'utilisateur selon le moteur de recherche en question. La recherche de syntagmes nominaux permet de rechercher une chaîne de caractères exacte. Par exemple, la question "moteur de recherche" ne vous retournera comme résultat que les documents contenant exactement la chaîne "moteur de recherche".

Il n'est pas besoin d'être un spécialiste des systèmes de recherche d'information pour se rendre compte des limites des systèmes booléens. En fait, leur fonctionnement interne se résume à une simple recherche de chaîne de caractères, que cette chaîne soit ou non syntaxiquement correcte. L'utilisation de requêtes booléennes peut rapidement devenir complexe. Et finalement, sur Internet, qui utilise ces opérateurs pour effectuer des recherches ? Les documentalistes certainement, les informaticiens peut-être, l'utilisateur ordinaire sûrement pas !

***La recherche par liste de mots***

La recherche par liste de mots (ou pseudo langage naturel) permet de s'affranchir d'utiliser un langage d'interrogation pour effectuer une recherche. Les requêtes sont donc plus simples à formuler. Certes plus simples, mais également plus imprécises. En effet, ce mode d'interrogation souvent proposé par les moteurs (*AliWeb, AltaVista, EuroFerret, Excite, Galaxy, HotBot, InfoSeek, Lycos, Magellan, OpenText, WebCrawler, WWWWorm*) en complément de la recherche booléenne n'est en fait qu'une surcouche logicielle de cette dernière. La requête de l'utilisateur est retranscrite par le système en une expression booléenne suivant un schéma précis préétabli (AND implicite, OR implicite, troncature droite implicite, etc.).

En somme, même si effectuer une recherche à l'aide de tels systèmes est a priori plus simple que de l'effectuer à l'aide d'une requête booléenne, il est néanmoins nécessaire que l'utilisateur connaisse les traitements de reformulation de la question effectués par le moteur. Il pourra ainsi adapter sa requête au fonctionnement interne de ce dernier et il aura une meilleure compréhension des résultats obtenus.

***La recherche en langage naturel***

La recherche en langage naturel, de loin la mieux adaptée au texte, va permettre à l'utilisateur de formuler une question totalement libre (en langage naturel c'est-à-dire le

langage commun de "tous les jours") pour effectuer sa recherche. Une telle recherche nécessite une indexation et une recherche "intelligente" mettant en œuvre des modules de traitements linguistiques élaborés. Aucun système de recherche sur Internet ne dispose à l'heure actuelle d'une telle puissance de traitement du langage.

Les moteurs de recherche sont basés dans leur majorité sur la recherche booléenne, comme nous l'avons évoquée plus haut. Celle-ci repose sur la correspondance exacte entre la question et les mots clé indexés. Un des problèmes évidents de cette approche est de ne pas retourner une mesure de pertinence. Il existe, par ailleurs, un autre modèle appelé ***modèle espace vectoriel***, basé un système pondéré, permettant de mesurer la similarité entre la requête et les documents.

## 2.2 Modèle espace vectoriel

Le modèle espace vectoriel introduit par Salton et al. (1974) a été très peu utilisé par les moteurs de recherche, puisque un de ses problèmes majeurs réside dans le fait que ses performances sont fortement liées à la topographie de l'espace créé lors de l'indexation, comme le mentionne Salton lui-même. Néanmoins, ce modèle a été remis en avant de la scène dans la communauté de la recherche d'information avec l'arrivée des systèmes de recherche évolués tels que les agents et les systèmes multi-agents.

Dans le modèle espace vectoriel, le schéma figuratif de base proposé est le vecteur (Salton et McGill, 1983). Ainsi, dans cette représentation, les documents et les requêtes sont considérés comme un ensemble de vecteurs dans un espace à $m$-dimensions constitué de $n$ documents $Doc_i$ de la source d'information $S$, identifiés par $m$ termes d'indexation $T_j$. Les termes peuvent êtres pondérés suivant deux méthodes : discrète (par exemple 0 : le document ne contient pas le terme, 1 : le document contient le terme) ou continue (par exemple, selon l'importance du terme d'indexation dans le document, $T_j$ prendra une valeur plus ou moins importante entre 0 et 1). Un espace d'indexation à trois dimensions a l'aspect illustré à la Figure 2.4.

Nous représentons alors le i$^{ème}$ document $Doc_i$ ($i$={1, 2, ..., $n$}) par un vecteur à $m$ dimensions :

$VectDoc_i = (d_{i1}, d_{i2}, ..., d_{im})$,

où $d_{ij}$ représente le poids du j$^{ème}$ terme du vecteur $VectDoc_i$.

Pour effectuer une recherche dans un espace à $m$ dimensions composé de $n$ documents, il suffit donc de modéliser la requête $Q$ de l'utilisateur par un vecteur $VectQ = (r_1, r_2, ..., r_m)$ où $r_j$ représente le poids du j$^{ème}$ terme du vecteur requête $VectQ$ et de calculer chaque degré de proximité $s(VectQ, VectDoc_i)$.

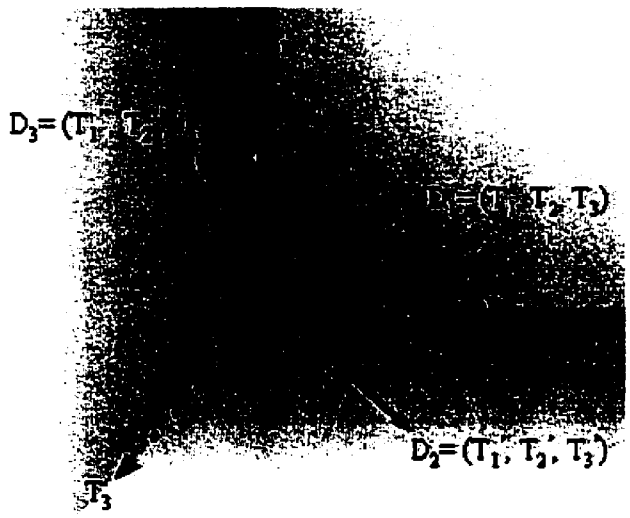


**Figure 2.4 Représentation vectorielle d'un espace de documents**

### 2.2.1 Métriques de proximité

Soient deux vecteurs de l'espace représentant un document $Doc_i$ et une requête quelconque donnée $Q$. À partir de ces vecteurs, nous pouvons calculer un coefficient de proximité $s(VectQ, VectDoc_i)$ entre $VectQ$ et $VectDoc_i$. Le coefficient de proximité obtenu $s(VectQ, VectDoc_i)$ prend en compte la similarité des termes utilisés dans les deux documents $Doc_i$ et $Q$, tout en tenant compte du poids des termes dans chacun d'eux ($d_{ij}$, $r_j$). Il existe plusieurs métriques permettant de déterminer la similarité entre ces deux vecteurs. Une première est définie comme le cosinus de l'angle entre les deux vecteurs

de l'espace. Cette mesure de la similarité est largement utilisée dans les systèmes de recherche, en raison de sa simplicité et de son efficacité (Salton et McGill, 1983). Ainsi, quand les termes d'indexation sont identiques dans les deux documents, l'angle est nul et donc la mesure de proximité est maximale. Une deuxième métrique, basée sur la distance euclidienne, est encore très populaire et utilisée pour mesurer la similarité entre les vecteurs dans l'espace vectoriel en question (Myaeng et Korfhage, 1990).

Quelle que soit la métrique choisie, la similarité entre deux vecteurs dans l'espace vectoriel sera inversement proportionnelle à la distance qui existe entre les points représentant ces vecteurs dans l'espace (Korfhage, 1997). De plus, n'importe quelle métrique choisie doit satisfaire les trois propriétés d'une distance : elle est positive, symétrique et respecte l'inégalité triangulaire.

Ainsi, toute fonction qui respecte ces trois propriétés peut être considérée comme une métrique pour la mesure de la similarité. Par exemple, si $d$ représente une certaine distance, alors une fonction définie par $e^{-d}$ peut être utilisée pour la mesure de la similarité. Finalement, si la distance entre deux points documents est égale à 0, alors la similarité entre ces deux documents sera égale à 1, et tous les autres documents dans l'espace vont y avoir des similarités plus petites ou égales vis-à-vis les deux documents en question.

Finalement, parmi les métriques qui ont été définies dans ce contexte, la famille des métriques $L_p$ généralise la métrique euclidienne (Korfhage, 1997). Cette famille de métriques est définie par l'équation suivante :

$$L_p(VectDoc,VectQ)=\left[\sum_{i=1}^{m} |d_i - r_i|^p\right]^{1/p} \qquad (2.1)$$

Dans cette équation, *VectDoc* désigne un certain document et *VectQ* la requête de l'utilisateur.

Comme cas particuliers de cette métrique, on a :

p = 1 qui réfère à une distance dans une dimension ; p = 2 qui désigne une distance euclidienne.

La distance euclidienne est la plus connue et la plus utilisée parmi les métriques. Notons ici que la métrique basée sur le cosinus de l'angle entre les vecteurs dans l'espace peut être considérée encore parmi les plus efficaces (Buckley, 1985 ; Kulyukin et al., 1996). L'équation (2.2) représente la mesure de la similarité, définie par Salton et McGill (1983), et basée sur le cosinus de l'angle entre un vecteur document *VectDoc* et un vecteur requête *VectQ* dans l'espace vecteur.

$$invCos(VectDoc, VectQ) = 1 - \frac{\sum_{i=1}^{m} d_i * r_i}{(\sum_{i=1}^{m} d_i^2 * \sum_{i=1}^{m} r_i^2)^{1/2}} \qquad (2.2)$$

### 2.2.2 Mise en paramètres de l'espace vectoriel

Plusieurs schémas ont été proposés dans le but de définir ou de «*paramétrer*» l'espace vectoriel, comme nous l'avons déjà mentionné. Salton et Buckley (1988) proposent une mise en paramètres très populaire et efficace pour représenter les vecteurs. Cette mise en paramètres, est appelée TF*IDF (Term Frequency * Inverse Document Frequency), où les composantes des vecteurs documents peuvent avoir des valeurs différentes de 0 et 1. Ce schéma de mise en paramètres est basé sur la fréquence d'occurrence des termes $T_j$ dans les documents de la source $S$. Ainsi, dans ce schéma, le vecteur document $VectDoc_i = (d_{i1}, d_{i2},..., d_{im})$, est défini tel que :

$$d_{ij} = \text{Occ}_{Tj}(VectDoc_i) * \log_2(s/\text{nb}_{Tj}(\text{documents})) \qquad (2.3)$$

où :

Occ $_{Tj}$ (D) = nombre d'occurrences du terme $T_j$ de $S$ dans le document $D$ ;

$s$ = nombre de documents dans la source d'information ;

$\text{nb}_{Tj}$ (documents) = nombre de documents contenant le terme $T_j$ dans la source.

Le modèle espace vectoriel, contrairement au modèle booléen, permet donc de pondérer les requêtes et les résultats. Il est possible de calculer un degré de similarité

entre les documents de la base, mais aussi entre une question et l'ensemble des documents.

## 2.3 Bibliothèques virtuelles

Les progrès récents dans la technologie des réseaux de télécommunications et la quantité gigantesque d'informations disponibles sur supports électroniques, ont donné naissance à une explosion de travaux de recherche dans le domaine des bibliothèques virtuelles (Singh, 1993 ; Benslimane et al., 1993 ; Wooldrige et Jennings, 1994 ; Maes, 1994 ; Lander et Lesser, 1994 ; Finin, 1993 ; Mayfield et al., 1995 ; Labrou et Finin, 1997).

L'élaboration d'une bibliothèque virtuelle revient en quelque sorte à reproduire, dans le cadre d'un réseau ou d'un système informatique, le modèle des bibliothèques classiques. Ainsi, une bibliothèque virtuelle est dotée d'assistants de recherche d'information qui mettent leur savoir-faire au profit du client utilisateur. Ayant accès à un ensemble de sources d'information, ces assistants de recherche virtuels, à l'image des bibliothécaires, jouent le rôle d'intermédiaires entre l'information disponible et les utilisateurs. Leur travail consiste, tout comme dans le monde réel, à faciliter la tâche de recherche d'information. Afin de jouer ce rôle d'intermédiaire, les bibliothécaires doivent savoir utiliser efficacement les sources d'information accessibles ; ils doivent aussi reconnaître les différents utilisateurs, cerner leurs besoins d'information ainsi que leurs intérêts afin de pouvoir répondre adéquatement à chacun d'eux. La bibliothèque virtuelle intègre donc à la fois l'information et les services offerts par les bibliothécaires. Nous considérons donc qu'une bibliothèque virtuelle est plus qu'un simple moteur de recherche électronique. Elle doit être vue comme un fournisseur actif et dynamique d'informations changeant rapidement (Paepcke et al. 1998).

Atkins et al. (1996) soulignent que les bibliothèques virtuelles doivent être basées, d'une part, sur un modèle universel qui prend en compte la spécificité de chacune des sources d'information et, d'autre part, sur une large gamme de services facilitant la

recherche d'information pertinente. Par ailleurs, Knoblock et Arens (1994) soulignent que la construction d'une bibliothèque virtuelle unique intégrant toutes les sources d'information est impossible. Ils proposent alors une approche qui consiste à décentraliser l'accès à l'information en créant des entités informatiques spécialisées appelées agents. Ces derniers sont responsables d'un sous-ensemble de l'information, ils communiquent et collaborent entre eux afin d'échanger des informations et des résultats partiels. De cette manière, chaque agent peut encapsuler un modèle détaillé de son domaine d'expertise, en bénéficiant des connaissances des autres agents. Cette approche a été largement soutenue dans la communauté de recherche sur les bibliothèques virtuelles et a donné naissance à plusieurs architectures basées sur le modèle agent et les systèmes multi-agents (UMDL, 1995 ; LIRA, 1995 ; MeDoc, 1996 ; ViSe, 1997 ; Syskill, 1997 ; ISAME, 2000).

## 2.4 Agents et les systèmes multi-agents

Les systèmes multi-agents constituent un des domaines de recherche les plus prometteurs pour la recherche d'information numérisée. Ce domaine bénéficie des résultats de la recherche dans de nombreux autres domaines, dont les systèmes répartis, l'intelligence artificielle distribuée, la représentation des connaissances, la modélisation de l'utilisateur et les réseaux de télécommunications. Dans cette section, nous définirons, en premier lieu, ce qu'on entend par un système multi-agent, puis nous explorerons les composants de base de ces systèmes, à savoir les agents, dont nous verrons la définition controversée et les attributs généraux. Finalement, nous nous intéresserons, en particulier, aux agents intelligents de recherche d'information numérisée.

### 2.4.1 Définition d'un système multi-agent

Un système multi-agent est un système informatique qui se distingue par son environnement logiciel et matériel, par ses éléments de base (ses agents) et par les mécanismes d'actions et d'interactions existant entre ces éléments et l'environnement en

question. L'environnement d'un système multi-agent est défini par une description détaillée de toutes les ressources matérielles (mémoire, ordinateurs, réseaux, etc.) et logicielles (bases de données, systèmes d'exploitation, bases de connaissances, etc.) mises à la disposition du système. Comme le sous-entend son nom, les éléments de base d'un système multi-agent sont les agents. Ces derniers sont les entités logicielles qui caractérisent, par leurs actions sur l'environnement et leurs interactions, l'activité principale du système. Finalement, les mécanismes d'actions des agents sur l'environnement et les mécanismes d'interactions entre les agents sont souvent régis par des règles préétablies du type événement-action et d'un langage de communication (KQML, KIF, etc.)

Un système multi-agent est généralement mis en place afin de résoudre ou de modéliser un problème ou une activité complexe. La représentation d'un problème par un système multi-agent consiste à distribuer les tâches entre plusieurs entités autonomes pouvant communiquer et collaborer entre elles afin d'exécuter la tâche globale à laquelle le système est dédié (Singh, 1993). Les fonctionnalités du système peuvent être considérées comme des super-tâches décomposables en tâches normales qui sont exécutables par des agents spécialisés. En règle générale, chaque tâche normale peut être réalisée par un seul agent, même si certaines d'entre elles requièrent la coopération de plusieurs agents. De plus, lorsqu'un ensemble de tâches est suffisamment homogène, il peut être accompli par un agent unique. Dès lors, cet ensemble de tâches peut être considéré comme une tâche unique complexe, décomposable en sous-tâches (Pierre et Hotte, 1996).

### 2.4.2 Définitions d'un agent

Le mot *agent* est actuellement très en vogue dans la presse informatique populaire comme dans celle des communautés d'intelligence artificielle (IA) et des sciences informatiques. Néanmoins, cette ferveur pour ce mot crée une ambiguïté quant à la définition précise de la notion d'agent. En effet, la raison pour laquelle il est difficile de définir précisément ce que sont les agents, c'est que ce terme est devenu une vraie

« couverture » de plusieurs corps de recherche et de développement hétérogènes en génie logiciel et en intelligence artificielle. D'ailleurs, pour souligner cette difficulté dans la définition de ce terme, Nwana (1996) déclare :

> *« Lorsque nous sommes amenés à le faire brièvement, nous définissons un agent comme étant un composant logiciel et matériel qui est capable d'agir avec précision afin d'accomplir des tâches au nom de son utilisateur. Lorsqu'on nous donne le choix, nous dirions plutôt que c'est un terme couverture, un méta-terme, ou une classe, qui chapeaute une panoplie d'autres types plus spécifiques d'agent, et nous poursuivons alors en énumérant et en définissant ce que sont ces autres types d'agent. De cette façon, nous réduisons les chances d'entrer dans les arguments philosophiques et stériles habituellement suscités par l'ancienne définition, dans laquelle n'importe quel vieux logiciel peut être remodelé en un logiciel orienté-agent.»*

La définition du dictionnaire de la langue française (*Le Petit Robert 1*) décrit un agent comme étant "une personne chargée des affaires, des intérêts, d'un individu, d'un groupe ou d'un pays, pour le compte desquels elle agit". Cette définition générale de l'agent met l'accent sur deux aspects fondamentaux :

- un agent est actif ;
- un agent agit à la demande de quelqu'un ou de quelque chose.

Dans le cadre des systèmes multi-agents, il existe des notions d'agent très différentes les unes des autres. Néanmoins, il est possible de trouver un compromis entre ces notions. En effet, par agent, nous entendons un acteur artificiel qui remplit une ou plusieurs tâches d'assistance. C'est aussi une entité virtuelle qui est capable d'agir sur son environnement, qui dispose d'une représentation partielle de cet environnement, qui peut communiquer avec d'autres agents dans un univers multi-agent, et dont le comportement est la conséquence de ses observations, de ses connaissances et de ses interactions avec d'autres agents (Benslimane et al., 1993). Cette définition de l'agent

ressemble à celle donnée par Ferber (1997) qui définit les agents comme des «entités informatiques», réelles ou abstraites, de complexité variable.

Alain (1996), quant à lui, définit un agent tout simplement comme un programme informatique autonome, qui se déclenche sans une requête directe de l'utilisateur, en fonction du contexte, pour effectuer une tâche précise. D'autre part, Wooldridge et Jennings (1994) proposent leur définition de l'agent en mentionnant qu'un agent est une composante matérielle ou logicielle qui agit sans l'intervention directe d'un humain ou d'une autre entité. Un agent peut donc interagir avec d'autres agents ; il est capable de percevoir son environnement et de réagir en temps réel aux changements qui y surviennent.

Pattie Maes (1994) nous donne deux manières pour définir le terme agent. La première définition est générale et vise à couvrir la globalité des disciplines dans lequel ce terme agent est utilisé ; la deuxième est plus spécifique au domaine de l'assistance de recherche automatisée ou intelligente. Ainsi, un agent apparaît comme un logiciel qui évolue dans un environnement complexe et dynamique. En ce qui concerne l'assistance intelligente, Maes définit un agent comme un assistant personnel pour l'utilisateur. Cet agent doit être engagé dans un processus coopératif avec l'utilisateur pour qui il doit exécuter des tâches. Lorsque certaines tâches complexes sont déléguées aux agents, ces derniers doivent être en mesure de rendre transparente à l'utilisateur la complexité de ces tâches. Un agent accompagne l'utilisateur dans son environnement afin de réduire la quantité de travail de ce dernier et pour lui fournir l'assistance nécessaire. Cette assistance se révèle de plus en plus efficace lorsque l'agent apprend les intérêts, les habitudes ainsi que les préférences de son utilisateur (Maes, 1994).

Bien qu'il existe une grande variété de types d'agents, notre étude s'intéresse tout particulièrement aux agents de recherche d'information numérisée, ceux-ci correspondent plus particulièrement à la seconde définition de Maes puisqu'ils accompagnent et assistent l'utilisateur lors de sa recherche d'information.

### 2.4.3 Attributs d'un agent

Les attributs principaux d'un agent sont : l'autonomie, l'aptitude sociale et la coopération, la réactivité, la pro-activité, la mobilité, la continuité temporelle, l'adaptabilité, la personnalisation, et finalement, le risque et la confiance.

***Autonomie***

Un agent peut opérer sans une intervention directe, humaine ou non (FIPA, 1996). Cela signifie qu'il doit avoir un degré d'autonomie vis-à-vis de son utilisateur. Autrement, si une telle autonomie n'est pas présente, un agent se réduira à une sorte de frontal, irrévocablement fixe, qui verrouille les actions de son utilisateur. Un agent plus autonome peut poursuivre un agenda indépendant de celui de son utilisateur. Ceci exige l'intégration des aspects d'action périodique, d'exécution spontanée et d'initiative, permettant ainsi à un agent de prendre des décisions et d'entreprendre des actions de préemption ou des actions indépendantes qui sont éventuellement bénéfiques pour l'utilisateur.

***Aptitude sociale et coopération***

Les agents peuvent interagir avec d'autres agents et des humains (FIPA, 1996). La coopération Utilisateur-Agent peut être décrite comme une sorte de formulaire de *collaboration* lors de l'élaboration d'un contrat. L'utilisateur spécifie les actions qui doivent être entreprises pour son compte, et l'agent spécifie ce qu'il peut faire pour fournir les résultats. Ceci est souvent mieux décrit par une conversation dans laquelle chacune des parties peut demander des questions à l'autre pour vérifier que ces deux parties sont d'accord. Ainsi, les deux parties interagissent davantage comme des pairs (*peers*) dans les systèmes orientés-agents (Foner, 1993). La coopération inter-agents, quant à elle, est un mécanisme par lequel les agents échangent leurs connaissances, leurs croyances et leurs plans afin de travailler ensemble pour résoudre des problèmes qui dépassent leurs capacités individuelles.

***Réactivité***

Les agents peuvent percevoir leur environnement et répondre au moment opportun à des changements qui s'y produisent (FIPA, 1996). Leurs actions sont exécutées en résultat à des règles de déclenchement (*triggering rules*) ou à des plans d'exécution stéréotypés : mettre à jour la base des faits (ou l'espace de croyance) de l'agent, et envoyer des messages à d'autres agents de l'environnement (Chaib-Draa et al., 1996).

***Pro-activité***

Les agents peuvent dévoiler ou exhiber un comportement dirigé vers un but en prenant des initiatives (FIPA, 1996). Ils peuvent raisonner sur leurs intentions et leurs croyances, et planifier en conséquence leurs actions.

***Mobilité***

Cette aptitude caractérise uniquement les agents mobiles. Ceux-ci peuvent se déplacer vers d'autres environnements (FIPA, 1996). Ils peuvent transporter avec eux des données avec des instructions intelligentes qui peuvent être exécutées sur des sites distants.

***Continuité temporelle***

Les agents sont des processus en exécution continue, ils ne sont pas des programmes qu'on lance et qui se termine en un seul coup.

***Adaptabilité***

Les agents s'adaptent continuellement aux changements qui surviennent dans leur environnement.

***Personnalisation***

L'une des raisons d'être d'un agent est de permettre aux utilisateurs de réaliser de la meilleure manière possible certaines tâches. Puisque les utilisateurs ne font pas tous les mêmes tâches, et même ceux qui partagent les mêmes tâches le font de différentes façons, un agent doit être éducable. Idéalement, les agents doivent avoir des composantes d'apprentissage et de mémorisation (Foner, 1993).

***Risque et confiance***

L'idée d'agent est intimement liée à la notion de délégation. Nous ne voulons pas déléguer une tâche à quelqu'un ou à quelque chose si nous n'avons pas au moins une assurance raisonnable que l'entité à laquelle nous avons délégué peut effectuer la tâche voulue selon nos spécifications. Cependant, par définition, délégation implique un abandon du contrôle d'une tâche à une entité ayant des mémoires, des expériences et possiblement des agendas différents. Ainsi, en ne faisant rien par nous-mêmes, nous nous exposons à un certain risque qui consiste à la possibilité que l'agent puisse faire une erreur ou quelque chose de mauvais en regard du but visé. Ceci veut dire qu'il faut trouver un équilibre entre le risque d'erreur de la part de l'agent et la confiance qu'il puisse accomplir la tâche correctement (Foner, 1993).

### 2.4.4 Agent intelligent de recherche d'information

Un agent de recherche d'information numérisée, appelé aussi simplement agent d'information, est un agent spécialisé dans la recherche et la collecte des données numérisées. Ces agents ont accès potentiellement à une multitude de sources d'informations et sont capables de manipuler l'information collectée de ces sources afin de répondre à des requêtes formulées par des utilisateurs et d'autres agents (FIPA, 1996). Certains auteurs appellent ce type d'agent *Agents d'Internet* puisque la plupart des agents d'information opèrent sur des données présentes sur Internet.

Ce qui distingue un agent de recherche d'information d'un moteur de recherche par index, c'est que les agents d'information sont dotés d'autonomie, de capacité de communication avec d'autres agents ou de systèmes informatiques ou même avec des humains, et de capacités d'apprentissage.

Le mot "intelligent" est issu de travaux de recherche effectués en intelligence artificielle distribuée (Ferber, 1995). L'intelligence est rendue possible grâce à l'intégration des mécanismes d'apprentissage, de raisonnement et de planification dans les algorithmes de programmation (Harrison et al., 1995). Selon la normalisation, l'AFNOR définit l'agent intelligent en ces termes :

> "*Objet utilisant les techniques de l'intelligence artificielle : il adapte son comportement à son environnement et, en mémorisant ses expériences, se comporte comme un sous-système capable d'apprentissage : il enrichit le système qui l'utilise en ajoutant, au cours du temps, des fonctions automatiques de traitement, de contrôle, de mémorisation ou de transfert d'information.*"

## 2.5 Recherche intelligente d'information

Dans le contexte de ce mémoire, nous entendons par recherche intelligente d'information, une recherche qui prend en compte les besoins particuliers de l'utilisateur demandeur de cette information, pour délivrer en retour une information qui soit la plus fidèle à son domaine d'intérêt. Autrement dit, nous pouvons mesurer le degré d'intelligence d'une telle activité par la proximité entre le résultat d'une recherche entreprise et le profil personnalisé d'un utilisateur donné. En effet, il n'y a aucun doute que si on prend en compte les intérêts de l'utilisateur duquel provient une requête donnée, on peut efficacement améliorer la qualité des résultats de cette requête en fonction des besoins de ce dernier. Les questions qui se posent alors sont : Sous quelle

forme peut-on décrire le profil d'un utilisateur ? Et comment peut-on intégrer ce profil à une activité de recherche ?

De manière générale, un profil utilisateur consiste en un ensemble de spécifications et de caractéristiques décrivant un utilisateur particulier. Dans le cadre de la recherche d'information, ce profil peut représenter les besoins et les intérêts de l'utilisateur.

Myaeng Sung (1990), qui a publié plusieurs travaux sur l'aspect du *profil utilisateur*, définit trois méthodes pouvant servir lors de la recherche, afin d'utiliser les informations du profil : la modification de la requête par le profil, la prise en compte de la requête et du profil comme deux entités dirigeant la recherche et, finalement, le filtrage de résultats selon le profil.

***Modification de la requête par le profil*** : Cette méthode consiste à utiliser les informations du profil lors d'une étape de pré-traitement de la requête de l'utilisateur afin de créer une nouvelle requête pouvant décrire efficacement les besoins de ce dernier lors de sa recherche (cette méthode est appliquée avant d'entamer le processus de recherche).

***Requête et profil comme deux entités dirigeant la recherche*** : Cette deuxième méthode considère le profil et la requête de l'utilisateur comme deux entités séparées qui permettent de diriger la recherche d'information (elle est utilisée durant le processus de recherche).

***Filtrage de résultats selon le profil*** : Selon cette dernière méthode, les informations du profil utilisateur sont utilisées, dans une étape de post-traitement, pour faire filtrer et formater les résultats obtenus à partir de la recherche (elle est donc appliquée après le processus de recherche).

Nous présenterons, dans ce qui suit, trois approches basées sur le modèle espace vectoriel implantant respectivement les méthodes citées précédemment. Ces approches sont largement inspirées des travaux de recherche de Myaeng et Korfhage (1990), Korfhage (1984, 1988, 1991, 1997), Pazzani et al. (1995, 1997), Balabanovic et al. (1995). Chacune de ces approches a démontré beaucoup d'efficacité en termes de performance du système ainsi que de la qualité des résultats obtenus (Myaeng et Korfhage, 1990; Pazzani et al., 1995, 1997).

### 2.5.1 Modification de la requête selon le profil

Dans le modèle de représentation par espace vectoriel, le profil de l'utilisateur (caractéristiques et intérêts de celui-ci) est considéré comme un vecteur de termes. De plus, chaque terme appartenant au profil a un certain poids $p_i$ représentant l'intérêt de l'utilisateur pour ce terme. D'une manière similaire aux vecteurs documents décrits précédemment, le vecteur profil *VectP* est construit de la façon suivante : *VectP* = ($p_1$, $p_2$, $p_3$,..., $p_m$), où $p_j$ désigne le poids du terme $T_j$ de S dans le profil de l'utilisateur :

$$p_j = \begin{cases} \text{Poids}(T_j) & \text{si le terme } T_j \text{ appartient au profil} \\ 0 & \text{sinon} \end{cases}$$

Ce *vecteur profil VectP* représentera le profil de l'utilisateur et sera utilisé durant le processus de recherche d'information.

Nous considérons ainsi que le *vecteur source* d'information S de termes $T_j$, les *vecteurs documents* $VectDoc_i$ de la source S, le *vecteur requête VectQ* ainsi que le *vecteur profil VectP* de l'utilisateur sont déjà disponibles. Ainsi, les informations suivantes doivent être donc disponibles avant de commencer le processus de recherche :

S ($T_1$, $T_2$, $T_3$, ..., $T_m$) : l'ensemble des termes $T_j$ de la source d'information S ;

$VectDoc_i$ ($d_{i1}$, $d_{i2}$, $d_{i3}$,..., $d_{im}$) : vecteur poids des termes $T_j$ dans le document $Doc_i$ ;

où i = 1, 2, ..., n ;

*VectQ* ($r_1$, $r_2$, $r_3$, ..., $r_m$) : vecteur poids des termes $T_j$ dans la requête Q de l'utilisateur ;

*VectP* ($p_1$, $p_2$, $p_3$, ..., $p_{nm}$) : vecteur poids des termes $T_j$ dans le profil P de l'utilisateur ;

Dans cette première approche, appelée modèle d'interaction profil/requête, le but consiste à appliquer certaines modifications sur la requête avant de commencer la recherche d'information. Étant donné que l'utilisateur n'est pas toujours capable de bien préciser ses besoins d'information à partir d'une simple requête soumise au système, cette requête est souvent incomplète, non précise et ne décrit pas suffisamment les besoins de l'utilisateur. Il sera donc souhaitable de faire ajuster la requête selon les données du profil lors d'une étape de pré-traitement, avant même de commencer le processus de recherche. Dans la représentation espace vectoriel, ajuster la requête selon le profil correspond à faire approcher le point requête Q vers le point profil P dans l'espace, tel qu'illustré à la Figure 2.5. De cette manière, la nouvelle requête modifiée Q' sera utilisée lors de la recherche et peut influencer les résultats de cette recherche. Cette modification de la requête peut être faite de différentes manières. Ainsi, Myaeng et Korfhage (1990) proposent une simple modification linéaire de la requête vers le point profil par l'équation suivante :

$$VectQ' = k\ VectP + (1-k)\ VectQ \qquad (2.4)$$

VectQ désigne le vecteur requête originale ;

VectP le vecteur profil utilisateur ;

VectQ' le nouveau vecteur requête ;

k une valeur constante entre 0 et 1.

L'interprétation géométrique de cette équation montre évidemment que le point requête modifiée Q' sera sur le segment entre P et Q. Remarquons que, si k =0, alors la requête Q reste inchangée. Par contre, si k=1, alors la requête modifiée Q' sera remplacée par le profil P de l'utilisateur. Finalement, une valeur de k entre 0 et 1 correspond à une position intermédiaire de Q' entre P et Q.

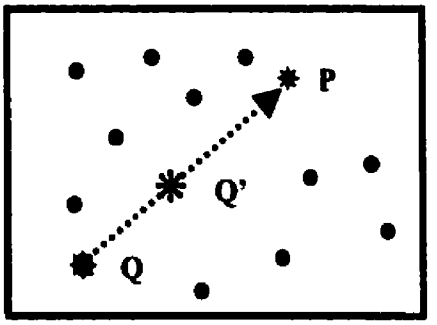


**Figure 2.5 Modification de la requête originale par le profil**

Une autre transformation ou modification de la requête selon le profil, nommé transformation linéaire par morceaux (Korfhage, 1997), semble plus efficace mais un peu plus complexe. Dans cette modification, les valeurs de composantes ou les poids des termes de la requête Q sont modifiés de différentes manières (selon leur apparition à la fois dans la requête et dans le profil). Ainsi, cet auteur définit quatre cas pour obtenir la nouvelle requête Q' modifiée selon le profil (rappelons que dans ce cas, S =($T_1$, $T_2$, $T_3$, ..., $T_m$) correspond à l'ensemble de tous les termes $T_j$ de la source d'information S). Ainsi, ces quatre cas sont définis comme suit :

- Si le terme $T_j$ de S appartient à la requête Q et au profil P, alors on fait une simple combinaison linéaire convexe, comme précédemment ;
- Si le terme $T_j$ de S appartient à la requête Q et n'appartient pas au profil P, alors on ne fait aucune modification sur la requête, ou bien on diminue un peu la valeur du terme $T_j$ dans la requête (de 5% par exemple), étant donné que le terme $T_j$ n'intéresse pas l'utilisateur ;
- Si le terme $T_j$ appartient au profil P et n'appartient pas à la requête Q, alors :
  a) ne pas introduire le terme $T_j$ dans le contexte de la recherche ($r'_j = 0$) ou,
  b) introduire le terme $T_j$ en diminuant son poids de moitié ($r'_j = 0.5 * p_j$ ), étant donné que le terme intéresse l'utilisateur ;
- Si le terme $T_j$ n'appartient pas à Q et n'appartient pas à P, alors $r'_j = r_j = 0$.

Notons que dans cette deuxième modification de la requête par le profil, il n'y a aucune garantie que la requête modifiée Q' sera sur le segment entre les deux points, requête originale Q et profil P, puisque chaque composante de Q est modifiée différemment. La nouvelle requête Q' ainsi obtenue est, évidemment, plus significative et nous pouvons nous servir de cette nouvelle requête lors du processus de recherche raffinée d'information.

### 2.5.2 Requête et le profil utilisateur comme deux entités séparées

Dans cette approche, deux modèles peuvent servir afin d'utiliser les information du profil pour diriger la recherche selon les besoins de l'utilisateur. Les deux modèles présentés dans cette section considèrent la requête originale Q et le profil utilisateur comme deux entités séparées. Ces deux modèles sont appliqués durant le processus de recherche d'information et correspondent donc à la deuxième méthode définie par Myaeng et Korfhage (1990). Ces modèles visent à retourner les documents qui correspondent à la requête ainsi que le profil de l'utilisateur à la fois. Les deux modèles présentés dans cette section sont : le modèle ellipsoïdal et le modèle ovale.

***Modèle ellipsoïdal***

Comme son nom l'indique, ce modèle consiste à trouver les documents $D_i$ pertinents à la requête Q ainsi que le profil P de l'utilisateur à la fois, en se basant sur la forme de l'ellipsoïde ayant les deux points requête Q et profil P comme foyers, et tel que montré à la Figure 2.6.

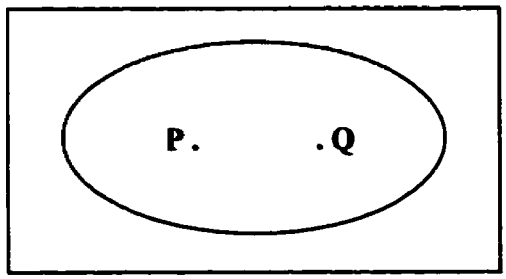


**Figure 2.6 Représentation du modèle ellipsoïdal**

L'équation de cette ellipsoïde est définie par :

$$\| VectDoc, VectP \| + \| VectD, VectQ \| = k$$

où :

$VectDoc = (d_1, d_2, d_3, \ldots, d_m)$ désigne un certain document de la source ;

$VectQ = (r_1, r_2, r_3, \ldots, r_m)$ requête de l'utilisateur ;

$VectP = (p_1, p_2, p_3, \ldots, p_m)$ profil de l'utilisateur ;

k : une certaine valeur constante prédéfinie (cette valeur k correspond à la taille de l'ellipsoïde) ;

$\| A , B \|$ : représente la distance entre les deux points A et B. Elle est mesurée avec une métrique quelconque.

De cette manière, chaque document $VectDoc_i$ de la source S tel que :

$\| VectDoc_i, P \| + \| VectDoc_i, Q \| \leq k$ (c'est-à-dire $VectDoc_i$ se trouve à l'intérieur de l'ellipsoïde) sera considéré comme pertinent à la requête et au profil de l'utilisateur. Notons ici que les composantes des *vecteurs* ou des *points* documents, requête et profil correspondent à des valeurs réelles ([-5, 5] par exemple) qui représentent les poids des termes $T_j$ de la source $S = (T_1, T_2, T_3, \ldots, T_m)$, dans un document Doc, dans la requête Q ainsi que dans le profil P de l'utilisateur.

***Modèle Ovale***

Dans le modèle ellipsoïdal, lorsque les deux points requête Q et profil P sont loin l'un de l'autre ou si la valeur de k est grande, alors la taille de l'ellipsoïde sera encore grande et, par conséquent, le nombre de documents retournés va augmenter aussi (tous les documents dans l'espace de l'ellipsoïde seront considérés comme résultats pertinents). Parmi ces documents retournés, on peut avoir des documents qui ne sont pas proches ni de la requête Q, ni du profil de l'utilisateur P, ce qui est indésirable dans ce contexte. Autrement dit, l'interprétation géométrique de cet ensemble de documents, considérés

non pertinents, est la partie de l'ellipsoïde entre les deux points requête Q et profil P de l'utilisateur (Figure 2.6).

Afin de surmonter cette faiblesse dans le modèle ellipsoïdal, le modèle ovale a été proposé. En effet, ce modèle basé sur la représentation géométrique de la forme de l'ovale, définie par Cassini, peut résoudre le problème grâce à la forme de cette ovale qui peut être changée selon la distance entre les deux points requête Q et profil P, et tel que montré à la Figure 2.7.

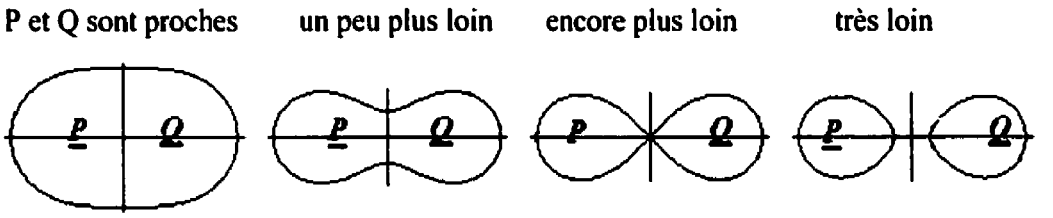


**Figure 2.7 Représentation du modèle ovale**

L'équation de cette ovale est définie par :

$\| VectDoc, VectP \| * \| VectDoc , VectQ \| = b^2$

où :

VectP et VectQ représentent les foyers de l'ovale ;

$b^2$ correspond à une certaine valeur constante positive ;

$\| VectP , VectQ \| = 2a$ et la forme de l'ovale dépend de $c = b / a$.

Ce modèle ou cette représentation géométrique de l'ovale répond exactement à ce qu'on vise dans ce contexte et qui permet de résoudre le problème du modèle précédent (ellipsoïdal). On suppose que les points profil VectP et requête VectQ représentent les « foyers » de l'ovale, et les documents recherchés représentent les points se trouvant dans l'espace à l'intérieur de l'ovale. De cette manière, un document donné sera considéré comme pertinent si et seulement si :

|| VectDoc, VectP || * || VectDoc , VectQ || ≤ $b^2$ (c'est-à-dire, si le document VectDoc se trouve dans l'espace de l'ovale). Notons que || || représente la distance entre les deux points dans l'espace. Cette distance peut être obtenue en utilisant n'importe quelle métrique, tel que décrit dans le modèle espace vectoriel. De plus, Myaeng (1990) a démontré que la distance euclidienne semble la plus appropriée et la plus performante dans le modèle ovale (dans le modèle ellipsoïdal par exemple, selon ce même auteur, la mesure de similarité basée sur le cosinus de l'angle entre les vecteurs dans l'espace est plus efficace).

### 2.5.3 Filtrage des résultats selon le profil

Cette dernière approche est appliquée dans une étape de raffinement de recherche et de filtrage des résultats. Ainsi, lorsque l'utilisateur reçoit les résultats, il donne ses évaluations sur ces résultats en affectant une valeur de pertinence $e_i$ à chaque document $D_i$ retourné comme résultat. Dans ce cas, le profil P sera alors mis à jour selon les évaluations $e_i$ de l'utilisateur sur les résultats obtenus. Ensuite, un autre cycle de recherche sera effectué avec le profil P' nouvellement créé. La mise à jour du profil P est effectué de la manière suivante :

$$Vect\,P' = VectP + \sum_{i=1}^{n} e_i * VectDoc_i \qquad (2.5)$$

où

$VectP = (p_1, p_2, \ldots, p_m)$ désigne le vecteur profil original ;

$VectDoc_i = (d_{i1}, d_{i2}, \ldots, d_{im})$ le vecteur document d'un certain document $Doc_i$ ;

$e_i \in [-5, 5]$ la valeur entière qui représente l'évaluation donnée au document $Doc_i$ ;

$VectP' = (p'_1, p'_2, \ldots, p'_m)$ le vecteur profil modifié selon les évaluations $e_i$.

Dans cette approche proposée par Balabanovic et. al. (1995), étant donné que l'utilisateur n'est pas capable de fournir les informations qui correspondent à son profil, la recherche commence avec un profil initial vide. Il est important de noter ici, qu'avec cette approche, l'utilisateur ne formule pas une autre requête auprès du système. Ainsi, le

profil P de l'utilisateur sera reconstitué et mis à jour à partir des évaluations $e_i$ de l'utilisateur sur les résultats obtenus après chaque cycle de recherche effectué. De cette manière, le profil de l'utilisateur sera créé automatiquement. De plus, la qualité des résultats sera sans doute améliorée, en fonction des intérêts et des besoins de l'utilisateur.

Nous verrons, au cours du prochain chapitre, les hypothèses, les considérations et les limites que nous avons établies, afin de construire notre modèle conceptuel d'un agent de recherche intelligente d'information sur Internet.

# CHAPITRE 3
# MODÈLE CONCEPTUEL D'UN AGENT DE RECHERCHE

Un modèle conceptuel consiste à identifier et définir d'une manière abstraite les différents éléments et concepts représentant le fonctionnement d'un système donné. Élaborer un modèle conceptuel pertinent requiert l'application d'une méthode formelle d'analyse et de conception. Durant les cinq dernières décennies, les méthodes structurées et fonctionnelles telles que SA, SADT, et SART (Jaulent, 1990), ont laissé progressivement la place aux méthodes orientées objet, telles que MERISE, Booch, OMT ou encore OOSE (Desfray, 1996; Booch, 1996; Rumbaugh et al., 1991). Ces dernières ont rapidement convergé vers un langage de modélisation standard et unifié appelé UML (Muller et Gaertner, 2000), présenté pour la première fois en 1996, et en constante évolution depuis cette date. UML se définit comme un langage de modélisation graphique et textuel destiné à comprendre et décrire des besoins, spécifier et documenter des systèmes, esquisser des architectures logicielles, concevoir des solutions et communiquer des points de vue. Dans ce chapitre, nous allons fournir les différents diagrammes UML autour desquels s'articule notre modélisation d'un Agent de Recherche Intelligente d'Information sur Internet (ARIII). Nous y fournirons de plus les différents algorithmes et principes couvrant l'essentiel des activités de cet agent.

## 3.1 Étude préliminaire du modèle

UML se définit comme un langage de modélisation, plutôt que comme une méthode de modélisation. Il ne définit pas la démarche de modélisation, chaque concepteur étant libre de choisir le processus qui lui semble le plus adapté en fonction du type des systèmes ou des applications développées. Cependant, dans le cadre de la modélisation

d'une application informatique, les concepteurs d'UML préconisent d'utiliser une démarche (communément connue sous le nom du processus unifié UML) : itérative et incrémentale, guidée par les besoins des utilisateurs du système, et centrée sur l'architecture logicielle.

***Itérative et incrémentale*** : Le processus unifié UML doit se doter d'étapes d'avancement à différents niveaux d'abstraction allant, d'une manière itérative et incrémentale, du niveau le plus général à celui le plus détaillé.

***Guidée par les besoins des utilisateurs du système*** : Avec UML, ce sont les utilisateurs qui guident la définition des modèles. En effet, le périmètre du système à modéliser est défini par les besoins des utilisateurs.

***Centrée sur l'architecture logicielle*** : L'architecture logicielle décrit des choix stratégiques qui déterminent en grande partie les qualités du logiciel (adaptabilité, performances, fiabilité, etc.).

Dans cet ordre d'idée, nous allons commencer notre étude par un recueil préliminaire des besoins fonctionnels et opérationnels de notre système, puis nous allons identifier les acteurs et finalement les messages échangés entre le système et les acteurs. Nous raffinerons par la suite notre analyse et notre conception jusqu'à obtention de nos modèles dynamiques et statiques de l'agent ARIII.

### 3.1.1 Recueil préliminaire des besoins fonctionnels et opérationnels

Comme le préconisent les concepteurs d'UML, le point de départ de tout processus de modélisation est la capture préliminaire des besoins fonctionnels et opérationnels du système en question, en l'occurrence notre agent ARIII. Il s'agit ainsi de spécifier clairement les limites d'un tel agent, en décrivant son "savoir-faire" vis-à-vis de ses utilisateurs.

Notre étude préliminaire des besoins est établie en se basant essentiellement sur les principaux objectifs que nous avions énoncés au chapitre 1. Dans ce contexte, nous

établissons en premier lieu une liste des besoins fonctionnels de l'agent ARIII, suivie par une liste de ses besoins opérationnels.

## A. Besoins fonctionnels

### A1. *Parcours d'Internet et collecte des documents*

La raison d'être de l'agent ARIII est de pouvoir répondre aux requêtes d'information sur Internet formulées par des utilisateurs, à l'image de ce que font les moteurs de recherche traditionnels, tout en apportant en plus une dimension "intelligente" à une telle activité en prenant en compte le profil de chacun de ces utilisateurs lors du filtrage des résultats. Ainsi, il est évident qu'un tel agent doit être capable de parcourir les ressources d'information sur Internet, typiquement les pages du WEB, et d'en collecter les documents et les informations afin que celles-ci soient traitées et stockées préalablement à toute activité de recherche. L'étendue d'un tel parcours et d'une telle activité de collecte est déterminée par l'algorithme utilisé à cet effet et par les paramètres d'entrée d'un tel algorithme. En effet, c'est à l'administrateur d'un tel agent ARIII de fixer ces paramètres en fonction des performances et des besoins opérationnels de l'agent ; ainsi, il contrôlera le volume des données, l'étendue d'une telle collecte et la fréquence d'une telle activité. Comme nous l'avons soulevé lors des chapitres précédents un système de recherche efficace doit prendre en compte impérativement la nature dynamique et perpétuellement changeante de l'Internet et de son contenu. Ainsi, l'activité de parcours et de collecte d'informations est une fonction essentielle, récurrente et périodique dans un tel système.

### A2. *Traitement des documents collectés*

Une fois les documents collectés, ils seront traités et stockés préalablement à toute activité de recherche. En effet, cette fonction est une préparation à la recherche, il s'agit essentiellement d'extraire et de filtrer les données à partir des documents collectés sur Internet et de les mettre sous un format approprié à l'algorithme de recherche

d'information utilisé. Cette fonctionnalité correspond à l'activité d'indexation que font la plupart des moteurs de recherche sur Internet. Les documents ainsi traités forment, à l'issue de cette étape, une base de documents de l'agent. Notons ici qu'un certain nombre d'ouvrages et d'articles appellent par abus de langage une telle base de documents la base de données du système, ceci sous-entendant que l'ensemble des données traitées est généralement stocké dans une base de données. Nous ne retenons pas cette appellation puisque, dans la plupart des cas, il ne s'agit que de grands fichiers volumineux indexés mais certainement pas d'une base de données munie d'un schéma de données et d'un langage de définition et de manipulation de ces données. Ainsi, nous parlerons de la base de documents collectés plutôt que de la base de données collectées. L'activité de traitement des documents collectés en est une qui doit être automatisée et ne requiert aucune intervention externe. Le seul élément déclencheur d'une telle activité est l'activité de mise à jour de la base de documents existante à la fin d'une activité de parcours d'Internet et de collecte de nouveaux documents. Cette activité fait partie du cycle principal de mise à jour de la base de documents de l'agent.

**A3. *Gestion de profil des utilisateurs***

Notre modèle conceptuel de l'agent ARIII définira et décrira en effet la classe d'agents ARIII. Nous pouvons en effet instancier à partir de cette classe différents agents, chacun pouvant être spécialisé dans un domaine donné. Ainsi, nous pouvons avoir un agent ARIII spécialisé en Mathématique et un autre spécialisé en Informatique. L'objectif principal d'une instance de l'agent ARIII consiste à répondre aux requêtes d'information de l'utilisateur en tenant compte des propres besoins de l'utilisateur en question. Autrement dit, l'ensemble de réponses retourné par l'instance de l'agent ARIII à une requête d'un utilisateur $U_x$ est filtré et délimité selon le profil $PU_x$ que possède cette instance de l'utilisateur $U_x$. Il n'existe pas une définition universelle du profil d'un utilisateur. Néanmoins, on peut définir pour les besoins de notre modèle ce que l'on entend par le profil de l'utilisateur : le profil $PU_x$ d'un utilisateur $U_x$ donné est défini

comme un ensemble de mots clés pondérés décrivant l'utilisateur $U_x$ en terme de ses préférences, besoins, intérêts et connaissances. Ces informations permettent à l'agent de mener la recherche conséquente à une requête R donnée selon le profil $PU_x$ de l'utilisateur $U_x$ qui est à l'origine de cette requête. Une telle démarche de personnalisation de la recherche rendra les résultats plus précis et certainement plus spécifiques aux attentes de l'utilisateur en question. L'agent permet à tout utilisateur proprement inscrit et identifié auprès de lui, de créer, modifier, activer et désactiver son profil. L'utilisateur peut ajouter à son profil des nouveaux mots clés ou même retirer des mots clés existants. Il peut aussi changer, selon son propre jugement, le poids d'un mot clé dans son profil. Lors de la création d'un profil, l'utilisateur pourra choisir un profil parmi un certain nombre de profils types préétablis comme base par l'administrateur de l'agent. Une fois choisi, un profil peut être personnalisé par l'utilisateur selon ses propres préférences. L'utilisateur peut à tout moment modifier son profil, il peut aussi décider de se retirer du système en effaçant son profil. La gestion des profils des utilisateurs de l'agent est une tâche essentielle du système. Elle devra fournir aux utilisateurs de l'agent ARIII un point initial où ils peuvent exprimer leurs besoins informationnels. L'agent ARIII doit fournir la ou les interfaces nécessaires qui permettent à l'utilisateur de créer et de mettre à jour son profil.

**A4. *Création et exécution d'une requête (activité de la recherche d'information)***

Le besoin fonctionnel le plus crucial à notre agent ARIII est en effet la fonction de recherche d'information qu'il fournit à ses utilisateurs. L'activité de recherche de notre agent est au cœur de l'architecture de celui-ci. Un utilisateur $U_x$ possédant un profil $PU_x$ formule auprès de l'agent une requête R. Celle-ci prend la forme d'un ensemble fini de mots clés pondérés par l'utilisateur. Les données provenant de la requête R sont par la suite utilisées conjointement avec ceux du profil $PU_x$ afin d'effectuer une recherche d'information sur la base de documents de l'agent en question. L'agent ARIII doit en premier lieu fournir l'interface qui permet à l'utilisateur d'exprimer sa requête et

d'intégrer en deuxième lieu un algorithme de recherche d'information dans la base de documents.

**A5.** ***Présentation des résultats d'une requête***

Ce besoin fonctionnel correspond à la nécessité d'organiser les résultats en fonction de la pertinence allouée par l'algorithme de recherche aux documents retournés, selon la requête et le profil de l'utilisateur. Un classement par ordre décroissant de pertinence sera utilisé.

**A6.** ***Collecte de la réaction de l'utilisateur (feedback)***

Cette fonction correspond à cueillir la réaction de l'utilisateur face aux résultats qui lui sont présentés à l'issue d'une requête. L'utilisateur allouera volontairement à chacun ou à quelques-uns des documents retournés un pourcentage de satisfaction. Ce dernier pourra être utilisé afin de raffiner la recherche, et un document ayant un degré de satisfaction dépassant un seuil établi peut être considéré comme document clé lors du prochain cycle de parcours et de collecte de documents sur Internet.

**A7.** ***Suggestion et raffinement***

Une fois que l'utilisateur a fourni sa réaction aux documents retournés, l'agent peut lui suggérer de raffiner la requête en utilisant les données cueillies suite à sa réaction. Il s'agit donc de relancer le processus de recherche après que l'agent ait raffiné la requête en y apportant des nouveaux paramètres. Cette fonction peut être utilisée successivement jusqu'à satisfaction de l'utilisateur.

**A8.** ***Administration de l'agent***

Bien que l'agent possède un degré d'autonomie qui lui permet d'effectuer pour le compte d'un utilisateur donné la recherche et le filtrage d'information sans une intervention directe de ce dernier aux différentes étapes de ces activités, il reste indispensable à un tel

agent de posséder un ensemble de mécanismes et de services qui permettent d'administrer son comportement global et de contrôler les paramètres essentiels de son fonctionnement global. Ainsi, chaque instance de l'agent ARIII doit avoir un administrateur qui veille à gérer ses différents paramètres et à garantir ainsi sa performance. L'agent ARIII doit aussi fournir la ou les interfaces qui permettent à l'administrateur de gérer les paramètres qui lui sont associés.

### A9. *Inscription de l'agent ARIII à un environnement multi-agent*

Dans la perspective de rendre notre modèle versatile et ouvert au besoin futur d'intégrer l'agent ARIII dans une architecture multi-agent, dans laquelle plusieurs instances de l'agent ARIII peuvent collaborer ensemble afin de servir leur propres utilisateurs, nous allons établir à travers les deux derniers besoins fonctionnels les lignes directives d'une telle intégration. La satisfaction de ces besoins permettra à notre agent ARIII de s'intégrer et d'évoluer dans une architecture plus globale, à l'image des architectures multi-agents de bibliothèques virtuelles telles que UMDL (Lander et Lesser, 1994), ISAME (Pelletier et al., 2000) ou autres.

### A10. *Collaboration avec des agents externes*

*Demande de service*

L'agent A peut demander l'assistance d'un agent pair B en prenant le rôle d'un utilisateur fictif $U_B$ de B, en utilisant lors de toute demande d'information le profil type alloué à cet effet par l'administrateur de B. Ce principe ressemble au mécanisme d'accès qu'offrent de nombreux systèmes qui permettent leur utilisation sous forme d'un visiteur (*guest*) avec certaines limitations de service. Ainsi, l'agent A acheminera sa requête à l'agent B en utilisant les mécanismes de communication qui sont en vigueur dans l'architecture multi-agent utilisée.

*Prestation de service*

En complément à une demande de service, l'agent A peut fournir de l'assistance à un agent pair B en lui prêtant l'identité d'un utilisateur $U_A$ avec un profil type. Ainsi l'agent A pourra recevoir une requête d'information provenant de B, à laquelle il acheminera à ce dernier, une fois déterminé, l'ensemble des résultats en utilisant les mécanismes de communication qui sont propres à l'architecture multi-agent utilisée.

Notons ici que la communication entre les agent A et B, qu'elle soit pour une demande ou une prestation de service, pourra se faire par l'intermédiaire d'un ou de plusieurs agents tiers. Nous faisons abstraction donc de ces mécanismes de communication qui peuvent différer beaucoup d'une architecture à une autre, et nous nous contentons de considérer ces communications établies sur des canaux virtuels.

**B. Besoins opérationnels**

**B1.** ***Identification et sécurité***

Pour des fins évidentes de sécurité et d'identification l'agent ARIII doit intégrer les mécanismes d'accès confidentiel aux services fournis. Ainsi, avant toute prestation de service un utilisateur est invité à s'identifier par un mot d'identification et un mot de passe fournis tous les deux par l'administrateur de l'agent. La saisie de ces paramètres permettent à l'agent d'identifier d'une manière non ambiguë l'utilisateur en question et lui donnera selon le privilège qui lui est associé accès à un certain nombre de services. Nous distinguons deux type de privilèges : le privilège utilisateur et le privilège administrateur. Le privilège utilisateur est associé aux utilisateurs ordinaires de l'agent et permet d'accéder à des services classiques tel que effectuer une recherche ou gérer leurs propres profil d'utilisateur. Tandis que, le privilège administrateur est associé aux gérants de l'agent et permet d'accéder à des services évolués permettant de gérer les accès à l'agent (gestion des utilisateurs), de mettre à jour les paramètres internes de

fonctionnement (réglages de performance *tuning*), de gérer l'état global de l'agent (activer /désactiver les services de l'agent), etc.

**B2. *Volume de données***

Il existe des contraintes évidentes d'espace mémoire associé à la base de documents de l'agent ARIII. Ainsi, le modèle sera doté d'un mécanisme de contrôle des paramètres de gestion de la mémoire qui spécifient, d'une part, la taille limite de la mémoire cache associée aux opérations usuelles de l'agent et d'autre part, la taille limite et l'allocation de l'espace disque pour stocker la base de documents de l'agent.

### 3.1.2 Identification des acteurs

Comme nous l'avons dit précédemment, notre modèle de l'agent ARIII s'articule autour des besoins des utilisateurs externes de l'agent. Ainsi, nous allons établir la liste des acteurs dans notre modèle et leurs fonctions respectives.

D'après le formalisme UML, un acteur représente l'abstraction d'un rôle joué par des entités externes qui interagissent directement avec le système étudié. Un acteur peut consulter et modifier directement l'état du système, en émettant et en recevant des messages éventuellement porteurs de données.

Nous avons établi quatre acteurs principaux dans notre modèle :

- Utilisateur :

  L'utilisateur, comme son nom l'indique, est la personne qui formule des requêtes d'informations auprès de l'agent. Il crée et gère ses propres profils et il active ou désactive un parmi ses profils pour les fins de la recherche. L'utilisateur peut éventuellement réagir aux résultats de la recherche.

- Administrateur :

L'administrateur se charge de gérer tous les paramètres de l'agent ainsi que l'accès aux services de ce dernier. Il crée les profils types de l'agent et veille spécialement sur l'activité de parcours et de collecte d'informations sur Internet. L'administrateur détermine à l'aide des différents paramètres mis à sa disposition les limites informationnelles de l'agent. Il veille à l'ajustement et au réglage de performance de l'agent (*tuning*) à l'aide de la mise à jour des paramètres de l'agent.

- Navigateur Internet (Web) :

Cet acteur représente le module externe d'accès et de navigation sur Internet qui permettra à notre agent ARIII de parcourir et de collecter les documents afin de construire sa base de documents. Cet acteur est une composante logiciel externe à notre agent, tel que "Internet Explorer", "Netscape" ou un autre logiciel de navigation du Web.

- Agent externe Pair :

L'agent externe pair est en effet un agent pair qui éventuellement demande et/ou fournit des services à l'agent en étude. Cet acteur n'est pas essentiel à notre modèle dans le sens qu'il n'est indispensable que si on met en œuvre l'agent ARIII dans un contexte d'une architecture multi-agent.

La Figure 3.1 résume l'ensemble des acteurs établis dans notre modèle et illustre la relation qu'ils ont avec l'agent ARIII.

## 3.2 Modélisation des besoins fonctionnels de l'agent ARIII

Les besoins fonctionnels de l'agent ARIII incluent le parcours d'Internet et la collecte des documents, le traitement des documents collectés, la gestion du profil des

utilisateurs, la création et l'exécution d'une requête, la présentation des résultats d'une requête, la collecte de la réaction de l'utilisateur, entre autres. Nous allons dans cette section modéliser ces différents besoins.

### 3.2.1 Parcours d'Internet et la collecte des documents

Afin de parcourir l'Internet et d'en collecter les documents qui formeront la base de toute recherche ultérieure, un agent ARIII sera doté d'un algorithme de parcours partiel et de collecte des documents (pages) du WEB . Comme nous l'avons expliqué lors du chapitre précédent, un tel parcours se résume à un parcours d'un graphe dont les nœuds sont des pages WEB et les liens sont des hyperliens entre ces pages. Nous avons donc opté pour un parcours en largeur.

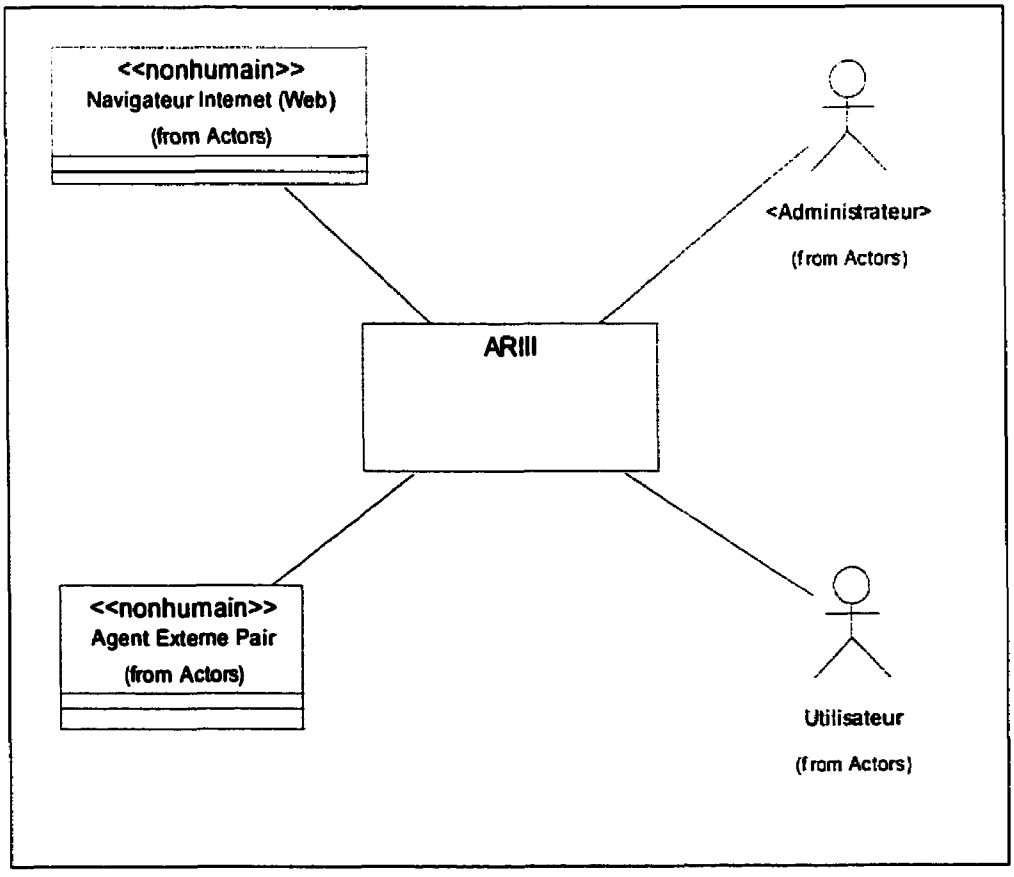


**Figure 3.1 Acteurs du modèle de l'agent ARIII**

L'algorithme reçoit comme entrées initiales un entier positif *iter* qui définit la profondeur voulue d'un tel parcours, et un ensemble initial S d'adresses URL (adresses de pages WEB) qui représente le point de départ du parcours et de la collecte. Ces deux paramètres sont considérés comme paramètres de l'agent et c'est à l'administrateur de ce dernier de préciser leurs valeurs. Éventuellement, l'administrateur, lors de la définition de S, peut puiser des adresses parmi celles des documents auxquels les utilisateurs ont alloué un degré de satisfaction élevé (par exemple > 90 %). L'algorithme de parcours partiel et de collecte des documents (pages) du WEB qui sera utilisé dans notre modèle est présenté dans la figure 3.2.

### 3.2.2 Traitement des documents collectés

Le traitement des documents collectés est une étape préalable à toute activité de recherche. Pour effectuer cette recherche, nous utiliserons le modèle espace vectoriel tel que défini au chapitre 2. Ainsi, la fonction de traitement des documents collectés intégrera tous les éléments permettant de formater les documents selon le modèle espace vectoriel. Le parcours des documents sur Internet conduit à établir un ensemble R des n documents collectés : $R = \{ Doc_1, ..., Doc_i, ..., Doc_n \}$. L'administrateur de l'agent ARIII a la charge d'établir une liste d'arrêt (*stoplist*) de termes, tels que « le, les, la, de, des, du, ou, et, au, dans, en, etc. », à exclure des documents collectés. La liste d'arrêt peut être associée à une langue donnée. Ainsi, nous pouvons écrire *stoplist* ∈ {*stoplistEnglish*, *stoplistFrench*, *stoplistSpanish*, etc.}.

Étape 1 : Création de la base de l'espace vectoriel.

L'ensemble R est traité pour en extraire les m différents termes $T_j$ n'appartenant pas à la liste d'arrêt *stoplist* qui constitueront la base B de notre espace vectoriel :

$$B = \{T_1, ..., T_j, ..., T_m\}$$

Les redondances de termes sont bien entendu à éliminer lors de l'établissement de la base B. Une telle base peut être améliorée en remplaçant la base d'origine par une base équivalente de termes préfixes $B_{pref}$ ou de synonymes $B_{syn}$ permettant de regrouper un ensemble de termes ayant même préfixe ou qui sont synonymes entre eux, et induisant toute la base B d'origine.

Étape 2 : Vectorisation des documents de R

Cette étape consiste à construire n vecteurs $VectDoc_i$ correspondant aux n documents $Doc_i$ de R dans la base B. Chaque vecteur $VectDoc_i$ possède m coordonnées ou poids $d'_{ij}$ correspondant au nombre d'occurrences du terme $T_j \in B$ dans le document $Doc_i$ :

$$\begin{cases} VectDoc_1 = (d'_{ij}, \ldots, d'_{ij}, \ldots, d'_{1m}) \\ \vdots \\ VectDoc_i = (d'_{i1}, \ldots, d'_{ij}, \ldots, d'_{im}) \\ \vdots \\ VectDoc_n = (d'_{n1}, \ldots, d'_{nj}, \ldots, d'_{nm}) \end{cases}$$

Étape 3 : Normalisation

L'ensemble de ces vecteurs constitue une matrice ($n \times m$) qui doit être normalisée afin que les poids correspondent à des nombres entiers compris dans l'intervalle [0,q]. q est un nombre arbitraire auquel nous avons choisi d'attribuer la valeur 5. Cette valeur va correspondre dans notre modèle à la gamme de pondération (entre 1 et 5) mise à la disposition de l'utilisateur lorsqu'il exprime l'importance qu'il alloue aux différents termes de sa requête ou de son profil. Pour réaliser la normalisation, il faut connaître le maximum des nombres d'occurrence des termes $d_{Max}$ parmi tous les vecteurs documents $VectDoc_i$ :

$$d_{Max} = Max\ (d'_{11}, \ldots, d'_{ij}, \ldots, d'_{nm})$$

**Action** PARCOURS_COLLECTE ( **Donnée** : *iter* entier positif,
**Donnée** : S Ensemble d'adresses URL,
**Résultat** : R Ensemble des documents collectés)

{

1. VISITE ← ∅; R ← ∅;
2. **si** S ← ∅ **alors** Finir.
3. ∀ $ad_i$ adresse URL appartenant à S, **enfiler** $ad_i$ dans FIFO*courant*; où FIFO*courant* est une file d'attente de type premier arrivé premier servi (*First In First Out*).
4. **tant que** *iter* +1 ≠ 0 **faire**
5. **tant que** FIFO*courant* ≠ ∅ **faire**
6. *ad* ← **Défiler** (FIFO*courant*); **si** *ad* ∉ VISITE **alors** Visiter l'adresse URL *ad*; **Collecter** le document *p* (page web) correspondant à l'adresse URL *ad*;
7. R ← R ∪ {p}; VISITE ← VISITE ∪ {*ad*};
8. ∀ $ad_i$ adresse URL contenu dans le document *p*, **enfiler** $ad_i$ dans FIFO*suivant* : FIFO*suivant* ← FIFO*suivant* ∪ {$ad_i$}; où FIFO*suivant* est une file d'attente de type premier arrivé premier servi (*First In First Out*).
9. **fin tant que**
10. *iter* ← *iter* – 1;
11. FIFO*courant* ← FIFO*suivant*;
12. FIFO*suivant* ← ∅;
13. **fin tant que**

}

**Figure 3.2 Algorithme de parcours partiel et de collecte des documents**

Le poids normalisé sera alors, pour chaque élément, calculé comme suit :

$$d^*_{ij} = 5 \times \left( \frac{d'_{ij} \times \log_2 \left( \frac{n}{nb_{Tj}} \right)}{d_{Max} \log_2 n} \right)$$

avec

$d'_{ij}$ : le nombre d'occurrences du terme $T_j$ dans un document $Doc_i$,

n : le nombre de termes de R,

$nb_{Tj}$ : le nombre de documents contenant le terme $T_j$ dans R,

la valeur 5 est choisie pour le paramètre q.

Cette normalisation du poids est une variation du modèle TF*IDF (*term frequency inverse document frequency*) vu dans le chapitre 2. La Figure 3.3 montre la tendance de la normalisation TF*IDF pour 100 documents. Nous voyons que cette normalisation donne plus d'importance aux documents lorsqu'ils sont peu nombreux à contenir le terme recherché. Notons que cette figure représente une tendance, la fonction n'est pas normalisée.

Notons ici que les propriétés suivantes sont vérifiées :

*Propriété 1* : $\forall j \in [1,m]$, si $T_j \in B$, alors $nb_{Tj} > 0$

*Propriété 2* : $\frac{n}{nb_{Tj}} \geq 1$

*Propriété 3* : $\forall i \in [1,n], \forall j \in [1,m], d'_{ij} \geq 0$

*Propriété 4* : $0 \leq d'_{ij} \times \log_2 \left( \frac{n}{nb_{Tj}} \right) \leq d_{Max} \log_2 n$

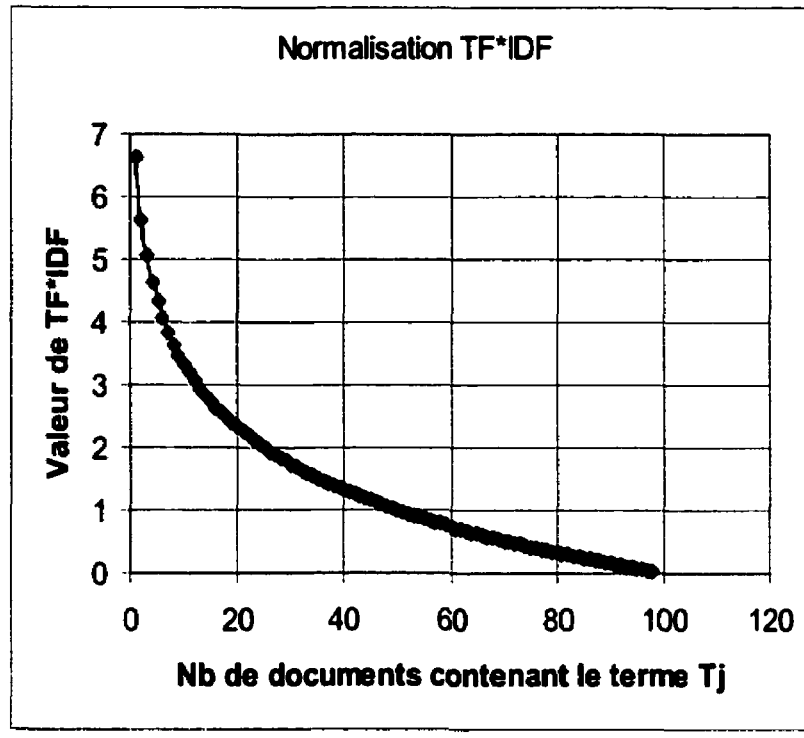


**Figure 3.3 Exemple de tendance TF*IDF pour n = 100 documents et $d'_{ij} = 1$ en fonction de $nb_{Tj}$.**

Nous choisissons d'utiliser des valeurs entières entre 0 et 5 effectuant une approximation par l'utilisation de la fonction par intervalle f : $d^*_{ij} \rightarrow d_{ij}$ définie comme suit :

$$\begin{cases} \text{si } d^*_{ij} = 0 \text{ alors } d_{ij} = 0 \\ \text{si } 0 < d^*_{ij} < 1 \text{ alors } d_{ij} = 1 \\ \text{si } 1 \leq d^*_{ij} < 1.5 \text{ alors } d_{ij} = 1 \\ \text{si } 1.5 \leq d^*_{ij} < 2.5 \text{ alors } d_{ij} = 2 \\ \text{si } 2.5 \leq d^*_{ij} < 3.5 \text{ alors } d_{ij} = 3 \\ \text{si } 3.5 \leq d^*_{ij} < 4.5 \text{ alors } d_{ij} = 4 \\ \text{si } 4.5 \leq d^*_{ij} < 5 \text{ alors } d_{ij} = 5 \end{cases}$$

Finalement, on remplace les termes $d^*_{ij}$ par les $d_{ij}$ ainsi normalisés pour obtenir la matrice des vecteurs normalisés suivante :

$$\begin{cases} VectDoc_1 = (d_{11}, \ldots, d_{1j}, \ldots, d_{1m}) \\ \vdots \\ VectDoc_i = (d_{i1}, \ldots, d_{ij}, \ldots, d_{im}) \\ \vdots \\ VectDoc_n = (d_{n1}, \ldots, d_{nj}, \ldots, d_{nm}) \end{cases}$$

### 3.2.3 Gestion de profil des utilisateurs

La gestion de profils des utilisateurs correspond à un ensemble de fonctionnalités qui permettent à un utilisateur de créer, modifier, activer ou désactiver ses profils :

*Création d'un profil* :

Pour faciliter la création d'un profil, un utilisateur $U_x$ peut utiliser comme point de départ un profil type parmi une liste mise à sa disposition par l'administrateur de l'agent. Les profils type appartenant à cette liste sont équivalents à des gabarits de profil (*profil templates*) généraux associés respectivement à des domaines donnés. Ainsi, nous pouvons avoir un profil nommé *Profil_Math* qui correspond à des utilisateurs s'intéressant aux mathématiques et un autre nommé *Pofil_IntelligenceArtificielle* s'intéressant au domaine de l'intelligence artificielle. L'utilisateur pourra modifier et personnaliser les paramètres de ces profils types pour exprimer et préciser ses propres besoins.

*Modification d'un profil* :

La modification d'un profil $PU_x$ consiste à lui ajouter un terme $T_k$ muni d'un poids $P(T_k)$ ou à lui retirer un terme $T_s$ (et par ce fait le poids qu'il lui est associé $P(T_s)$) ou à modifier le poids $P(T_m)$ attribué à un terme $T_m$ appartenant à ce profil.

*Activation d'un profil* :

Un utilisateur $U_x$ peut créer un ou plusieurs profils et les personnaliser comme bon lui semble, la principale contrainte que notre modèle impose est qu'un seul profil $PU_x$ parmi

l'ensemble de ces profils peut être considéré comme actif à un moment donné. Ainsi, un utilisateur est invité à activer un profil $PU_x$ afin qu'il soit utilisé lors de toute activité de recherche. Notons que, théoriquement, l'utilisateur peut créer un nombre infini de profils mais pour des raisons évidentes de contrôle d'espace et de gestion de l'agent l'utilisateur $U_x$ sera limité par un nombre maximal de profils $P_{max}(U_x)$ qui est un paramètre défini par l'administrateur de l'agent.

*Désactivation d'un profil* :

En complément à la fonction d'activation d'un profil, l'utilisateur peut décider de désactiver un profil pour activer un autre, ou tout simplement pour ne pas utiliser de profil lors de futures recherches.

### 3.2.4 Création et l'exécution d'une requête (activité de la recherche d'information)

*Création de la requête et la préparation du profil*

Une requête $QU_x$ de l'utilisateur $U_x$ est un ensemble de *w* termes $TQ_j$ pondérés avec des poids entiers $p(TQ_j)$ dont les valeurs sont comprises dans l'intervalle [1,5] :

$$QU_x = \{TQ_1, ..., TQ_j, ..., TQ_w\}$$

La requête $QU_x$ s'exprime dans la base $B = \{T_1, ..., T_j, ..., T_m\}$ par un vecteur :

$$VectQU_x = (r_1, ..., r_j, ..., r_m)$$

dont les poids sont déterminés comme suit :

$$\begin{cases} r_j = p(TQ_j) & \text{si } T_j \in QU_x \\ r_j = 0 & \text{sinon} \end{cases}$$

De même, un profil $PU_x$ de l'utilisateur $U_x$ est un ensemble de k termes $TP_j$ pondérés avec des poids entiers $p(TP_j)$ dont les valeurs sont comprises dans l'intervalle [1,5] :

$PU_x = \{TP_1, ..., TP_j, ..., TP_k\}$Le profil $PU_x$ s'exprime dans la base $B = \{T_1, ..., T_j, ..., T_m\}$ par un vecteur :

$$VectPU_x = (p_1, ..., p_j, ..., p_m)$$

dont les poids sont déterminés comme suit :

$$\begin{cases} p_j = p(TP_j) & \text{si } T_j \in PU_x \\ p_j = 0 & \text{sinon} \end{cases}$$

*Exécution d'une requête (activité de la recherche d'information)*

1. Combinaison de la requête et du profil :

   Il s'agit de la sommation des deux vecteurs, requête et profil.

   $$VectQ = VectQU_x + VectPU_x$$

   Si l'une des coordonnées de VectQ obtenue par cette sommation dépasse la valeur normalisée de 5, on la ramène à la valeur maximale 5.

2. Recherche des documents adjacents :

   Il s'agit de mesurer, pour i = 1, ..., n, les distances euclidiennes $d_i$ entre VectQ = $(q_1, ..., q_j, ..., q_m)$ et les $VectDoc_i = (d_{i1}, ..., d_{ij}, ..., d_{im})$ :

   $$d_i = \sqrt{\sum_{j=1}^{m} (d_{ij} - q_j)^2}$$

Comme nous l'avons vu au chapitre 2, le modèle vectoriel n'impose pas et ne privilégie pas l'utilisation d'une métrique donnée plus qu'une autre. Nous avons choisi d'utiliser une métrique Euclidienne, d'une part, pour sa popularité et sa simplicité en terme de calculs et de résultats intermédiaires, et d'autre part, pour sa performance démontrée par plusieurs publications (Myaeng et Korfhage, 1990 ; Korfhage, 1997).

### 3.2.5 Présentation des résultats d'une requête

Pour i = 1,..., m <u>si</u> $d_i \leq k$ <u>alors</u> l'agent présente le document $Doc_i$ (titre, URL, résumé) par ordre de proximité ($d_i$ - k).

Notons que k est un paramètre arbitraire de la recherche que l'administrateur de l'agent spécifie et règle selon la précision recherchée. Les résultats de la recherche sont des documents adjacents à la requête dans un rayon k (rayon d'adjacence ou de voisinage).

Lorsqu'on est dans un espace vectoriel à 3 dimensions, l'ensemble des résultats est formé des documents $VectDoc_i$ se trouvant dans une sphère de rayon k et de centre VectQ.

### 3.2.6 Collecte de la réaction de l'utilisateur (feedback), suggestion et raffinement

En réaction aux résultats retournés, l'utilisateur donne pour un certain nombre de documents résultats une valeur d'évaluation $e_i$ correspondant à un pourcentage entre 0 % et 100 % en fonction de la pertinence de chaque document.

Finalement, l'agent peut suggérer de relancer la recherche en reformulant la requête. Une nouvelle requête VectQ' est créée à partir de l'évaluation de l'utilisateur :

$$VectQ' = VectQ + \Sigma\, e_i \times VectDoc_i$$

Le processus de recherche est relancé à nouveau avec cette nouvelle requête. Cette fonction de suggestion et de raffinement peut être réitérée jusqu'à la satisfaction de l'utilisateur. Notons ici que, si l'agent ARIII évolue dans une architecture de coopération multi-agent, ce dernier pourra suggérer à cette étape à l'utilisateur le service d'un autre agent pair.

### 3.2.7 Inscription de l'agent ARIII à un environnement multi-agent

En faisant abstraction des mécanismes internes de communication (*blackboard, broadcasting, brokering agents, etc.*), des langages de communication (KQML, KIF, etc.), et des ontologies (*VirtualLibrary, Agents*, etc.) utilisés dans une architecture multi-agent, nous pouvons donner les règles minimales préconisées afin que l'agent ARIII puisse s'inscrire et évoluer dans un environnement de collaboration multi-agent.

Pour collaborer avec d'autres instances de l'agent ARIII, ou même d'autres agents dans un environnement multi-agent, il est indispensable qu'un agent soit identifiable d'une manière unique au sein de l'architecture. Ainsi, une instance de l'agent ARIII doit intégrer un mécanisme d'inscription auprès de l'architecture multi-agent en question.

Nous préconisons donc de définir la fonction *Inscription_ARIII*. Cette fonction inscrira une instance de l'agent ARIII à l'architecture en question en respectant les spécifications d'inscriptions propres à celle-ci et publiera le savoir-faire de l'agent. Dans le cas de l'agent ARIII, cette fonction publiera un profil type appartenant à la liste de profils types que l'administrateur de l'agent peut établir (*Profil_Math, Profil_Physique, etc.)*.

### 3.2.8 Collaboration avec des agents externes

Nous supposons que les agents A et B sont deux instances distinctes de ARIII. Nous supposons aussi que A et B sont inscrits auprès de l'architecture multi-agent utilisée.
Nous supposons dans ce qui suit que des mécanismes de délai de garde (*time-out*) et d'accusé de réception (*acknowledgement*) sont utilisés lors des différentes étapes de communications, nous faisons abstraction de ces mécanismes afin d'alléger le propos.

1. À l'issue d'une requête Q, l'agent A peut suggérer à son utilisateur $U_A$ de faire appel aux services d'un agent B.
2. L'agent A formule et envoie une demande de service auprès de l'agent B.
3. L'agent A attend la réponse de l'agent B.
4. Si l'agent B accepte favorablement la demande de A, alors B se met en état d'attente de la requête d'information provenant de A.
5. L'agent A envoie alors la requête Q à B.
6. B effectue la recherche sur sa propre base de documents en utilisant la requête Q et un profil type préétabli par l'administrateur de B pour toute prestation de service à un agent pair.
7. Finalement, B envoie les résultats obtenus à l'agent A qui, à son tour, les fera parvenir à l'utilisateur $U_A$.

## 3.3 Diagrammes du contexte dynamique et statique

Le diagramme UML du contexte dynamique consiste à modéliser le système en représentant d'une manière synthétique l'ensemble des messages (système ←→ acteurs) identifiés au cours de l'analyse des besoins. C'est donc un diagramme de collaboration entre le système en question et les acteurs qui seront en communication avec lui. La Figure 3.4 illustre le diagramme de collaboration exprimant le contexte dynamique de l'agent ARIII.

Le diagramme UML du contexte statique spécifie le nombre d'instances d'acteurs reliées au système à un moment donné. Ce diagramme est complémentaire au diagramme du contexte dynamique dans le sens qu'il veut mettre en évidence les différences qui existent en terme de multiplicité d'instances d'acteurs. La Figure 3.5 illustre le diagramme de classes exprimant le contexte statique de l'agent ARIII.

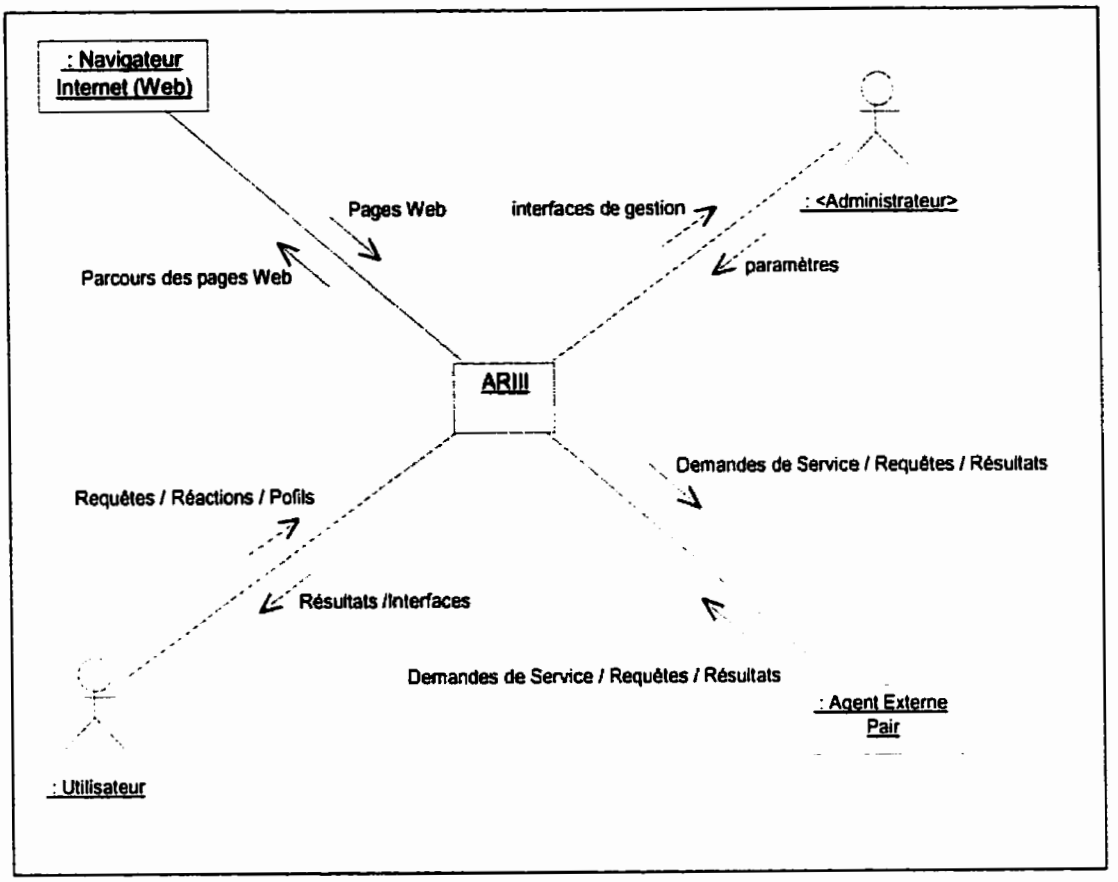


**Figure 3.4 Diagramme de contexte dynamique de l'agent ARIII**

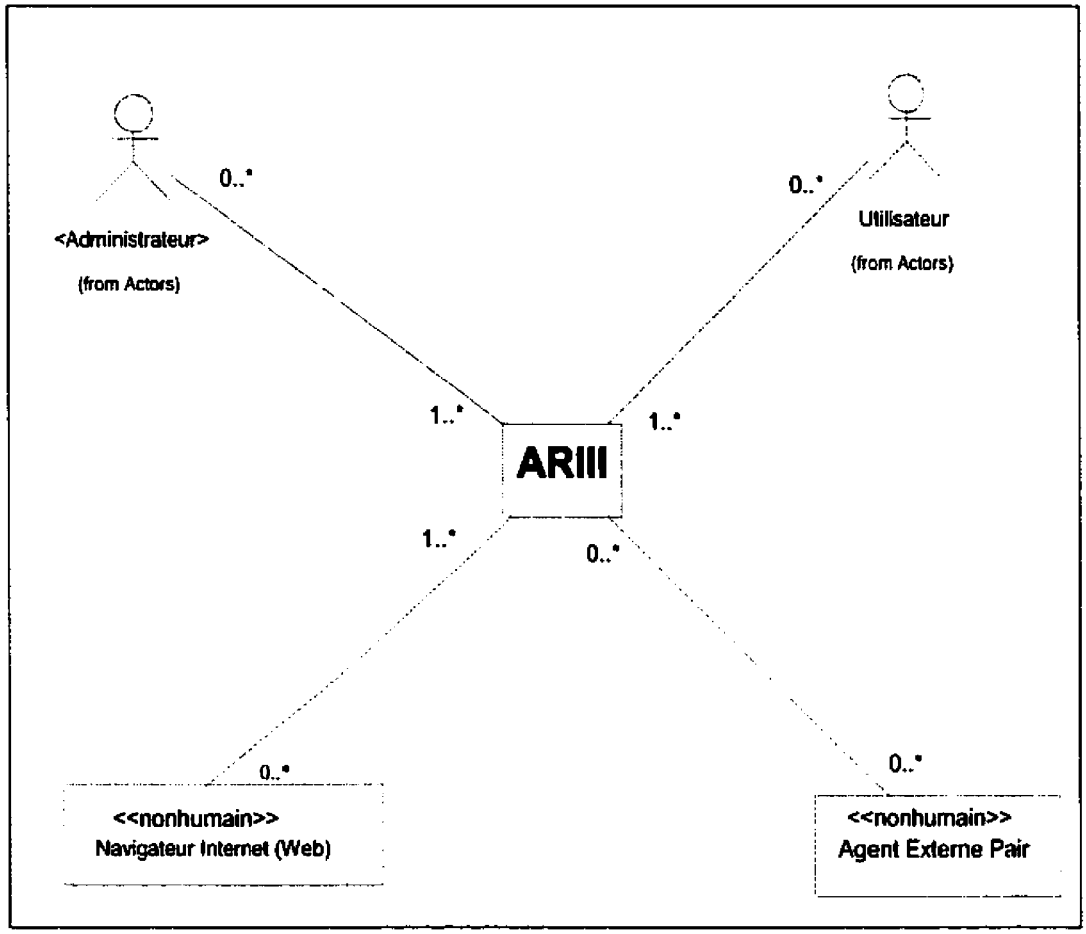


**Figure 3.5 Diagramme de contexte statique de l'agent ARIII**

# CHAPITRE 4
# CHOIX D'IMPLANTATION ET MISE EN ŒUVRE

Afin de démontrer la faisabilité et la viabilité du modèle conceptuel des agents ARIII présenté au chapitre précédent, une implantation partielle de ce modèle a été réalisé. Le présent chapitre décrit, dans un premier temps, les choix et les considérations technologiques qui ont servi de base à cette implantation. Dans un second temps, il présente le sous-ensemble du modèle sélectionné pour l'implantation. Dans un troisième temps, il présente l'aspect fonctionnel de cette implantation à travers des scénarios d'utilisation d'une instance de l'agent ARIII construit à l'aide de notre modèle. Finalement, le chapitre présente les résultats obtenus en terme d'évaluation comparative de notre prototype à un autre système connu de recherche d'information sur Internet.

## 4.1 Choix et considérations d'implantation

Bien souvent, lorsqu'il s'agit d'implanter et de mettre en place un modèle informatique, nous sommes confrontés au problème du choix technologique à faire pour réaliser ce modèle. En effet, le programmeur se retrouve, la plupart du temps, devant un ensemble de possibilités technologiques parmi lesquelles il doit faire un choix, bien souvent définitif, et dont toute remise en question ultérieure peut être extrêmement coûteuse. Ainsi, il est impératif, avant de se lancer dans le processus de la programmation et de la production du code, de faire un choix judicieux et éclairé parmi les technologies qui nous sont offertes. Cette étape est essentielle pour atteindre les normes qualitatives requises pour tout logiciel développé de nos jours.

Nous assistons malheureusement, dans la plupart des cas, à une omission délibérée dans la justification et la documentation des choix technologiques qui ont été entrepris préalablement à la phase de programmation. Cette omission est justifiée à tort par une

économie du temps et des coûts de développement. Cette fausse argumentation mènera tôt ou tard à des coûts et des dépenses prohibitifs lorsqu'il s'agira d'introduire de nouvelles extensions ou des mises à jour substantielles au logiciel en question. Ce n'est donc pas en vain, que les spécialistes et les professionnels du génie logiciel insistent sur l'importance de cette phase de choix technologique lors de la conception et sur la rigueur dans les décisions prises à ce niveau.

Bien qu'il s'agisse d'un prototype universitaire de recherche, notre modèle bénéficiera de toute notre attention quant à cette phase de choix technologique. Ainsi les sous-sections suivantes présenteront les choix et les considérations technologiques sur lesquels l'implantation partielle de notre modèle générique des agents d'information a été basée.

### 4.1.1 Choix du langage de programmation

L'implantation partielle de notre modèle générique des agents d'information a été réalisée en totalité à l'aide du langage de programmation JAVA. Ce choix est avant tout motivé par un ensemble de qualités intrinsèques qui distingue JAVA des autres langages et qui fait de lui un environnement de programmation idéal pour le développement des agents sur des supports architecturaux hétérogènes. Nous allons décrire en détail les différentes raisons pour lesquelles JAVA constitue cet idéal.

***Neutralité architecturale et portabilité***

L'aspect distribué est une caractéristique inhérente aux agents, et tout particulièrement aux agents d'information qui sont généralement répartis sur des réseaux hétérogènes. Ainsi, les applications doivent pouvoir être exécutées n'importe où sur le réseau sans avoir connaissance au préalable de la plate-forme matérielle et logicielle de la cible. JAVA fournit aux concepteurs et développeurs d'agents les avantages de la portabilité et de la neutralité architecturale. Pour résoudre ce problème de la distribution du code binaire, JAVA utilise un format de « code binaire » intermédiaire indépendant des architectures matérielles et des systèmes d'exploitation, le format de ce code binaire

intermédiaire étant architecturalement neutre. Le compilateur JAVA ne génère pas du « code machine » au sens d'instructions machine, mais plutôt un code intermédiaire (*bytecodes*) qui est un code indépendant du matériel et qui est destiné à une machine virtuelle implantée pour interpréter et exécuter ce code. Cet aspect de neutralité architecturale que nous venons de présenter est un grand pas franchi vers la portabilité. D'autre part, les fameux langages de développement C et C++ souffrent du fait que la désignation de nombreux types de données fondamentaux est dépendante de l'implantation matérielle sur laquelle le programme sera exécuté. JAVA élimine ce problème en définissant un comportement standard qui s'applique aux types de données, quelles que soient les plates-formes utilisées. JAVA spécifie la taille de tous les types de données primitifs et le comportement arithmétique qui s'y appliquent.

***Concurrence et multitâche***

À l'image du monde réel rempli d'événements qui se produisent en même temps, le monde des agents doit supporter la simultanéité des événements. Malheureusement, développer des programmes concurrents peut être beaucoup plus difficile et risqué que d'écrire un programme séquentiel (*single-threaded*) conventionnel avec un langage usuel. Le problème majeur rencontré lorsqu'on programme explicitement un support à la concurrence est qu'il subsiste toujours un risque d'interblocage entre processus, appelé aussi verrou mortel (*Deadlock*) à l'image de ce qui se produit dans la programmation de processus concurrents. Le support encastré de la concurrence est l'un des outils les plus puissants de JAVA, non seulement pour améliorer la performance interactive des opérations graphiques, mais aussi pour exécuter de multiples événements concurremment. La librairie JAVA fournit une classe appelée « *Thread.class* » qui supporte une riche collection de méthodes pour initier, exécuter, arrêter, et tester le statut d'un processus à poids léger (*thread*). Le support des processus à poids léger en JAVA inclut un ensemble sophistiqué de primitives de synchronisation basées sur le paradigme largement utilisé de la surveillance des conditions des variables. La plupart des architectures multi-agent utilisent la puissance de la programmation multitâche de

JAVA. Afin de garantir une continuité de service à l'agent ARIII et de doter son outil de recherche d'une capacité de parallélisme, la programmation multitâche de Java sera ainsi utilisée.

***Programmation orientée réseau***

Dans toute architecture multi-agent, les agents résidant sur différentes machines et environnements ont besoin de communiquer entre eux de l'information et de la connaissance sur leurs buts, leurs croyances et leurs intentions, afin de coordonner et de collaborer dans la recherche d'une solution cohérente. Cette communication constitue un aspect important dans le développement de tout système multi-agent. Le langage JAVA se prête spécialement bien à cet égard. En effet, il offre une librairie extensive de classes et de routines qui traitent facilement avec les protocoles UDP et TCP/IP, et qui supporte l'envoi, à travers un réseau, à la fois des messages diffusés (*Broadcasted messages*) et des messages adressés (*Directed messages*). De plus, JAVA intègre le concept d'appel aux méthodes d'objets distants appelés RMI (*Remote Method Invocation*) qui est d'une grande aide lorsque l'on veut créer une famille d'agents collaborants. En effet, ce concept RMI permet de faire appel aux méthodes d'objets distants sur réseau, ce qui remplace en quelque sorte la programmation parfois complexe des *sockets* dans une application client/serveur ou pair à pair.

***Sécurité***

Notre application est destinée à être intégrée dans un environnement distribué. JAVA est spécifiquement conçu pour être utilisé dans des environnements qui sont distribués et en réseau. À cette fin, une attention spéciale a été portée dès la conception sur l'aspect sécuritaire de JAVA. Les techniques d'identification sont basées sur des algorithmes d'encryptage puissants.

***Paradigme orienté-objet***

Notre modèle étant modulaire, il est logique qu'il soit implanté à l'aide d'un langage orienté-objet. Ce paradigme renforce et facilite la réutilisation et l'extensibilité des

modules. Parmi les langages de programmation orientée-objet, JAVA est l'un des rares qui renforce le concept orienté-objet. En effet, l'utilisateur n'a d'autre choix que d'encapsuler toutes les données dans des objets. Puisque les agents sont essentiellement construits autour de différents composants objets, JAVA est un langage idéal pour les développer.

***Facilité de connexion à l'Internet (spécifiquement le WEB)***

Comme le décrit notre modèle conceptuel, l'agent ARIII collecte et construit sa base de documents à partir d'un parcours de l'Internet (du WEB). JAVA offre un éventail de classes et d'outils de base facilitant l'accès, le parcours et la collecte de documents de l'Internet. En effet, les classes *URL* et *URLconnection* sont spécialement adaptées pour développer des applications qui ont besoin d'accéder aux contenus informationnels du WWW.

### 4.1.2 Choix de l'environnement de développement intégré (EDI)

Afin de développer une application en JAVA, il est possible d'utiliser l'environnement de développement de base JDK (*Java Development Kit*) que le groupe SUN, concepteur du langage JAVA, met gracieusement à la disposition des programmeurs. La dernière version du JDK (Java 2, version 1.2, 1999) de SUN offre certes une palette d'outils disponibles pour la gestion de projets, le déboguage, et l'aide, mais elle n'offre pas d'interface graphique (GUI) conviviale pour le développement des applications JAVA. Pour remédier à cette faiblesse, plusieurs grands groupes informatiques ont développé et commercialisé leur propre environnement de développement intégré (EDI) pour JAVA à l'image de ce qui se fait avec d'autres langages de programmation tels que C et C++. Nous retrouvons, parmi ces grands joueurs et vendeurs de plates-formes de programmation JAVA, les compagnies SYMANTEC avec son EDI nommé *Visual Cafe*, BORLAND avec son EDI nommé *JBuilder*, et MICROSOFT avec son EDI nommé *Visual J++*. Chacun de ces produits présente des qualités et des lacunes. Néanmoins *Visual Cafe* de SYMANTEC figure à la

tête du palmarès des EDI de JAVA, d'une part parce que les équipes de SYMANTEC étaient les pionniers dans le développement de EDI en JAVA, et par conséquent accusent d'une longueur d'avance devant leurs concurrents, et d'autre part parce que *Visual Cafe* et son ancêtre *Cafe* est le EDI le plus utilisé pour développer de grandes applications très connues en JAVA.

Bien que tous ces produits soient issus de différents groupes concurrents, ils se basent tous sur des versions du JDK développées et mises à jour continuellement par SUN. Ainsi, toute application développée à l'aide de l'un ou l'autre des EDI doit être interprétable et exécutable sur n'importe quelle plate-forme indépendamment de l'EDI qui l'a créée.

Notre choix d'EDI pour développer notre modèle en JAVA s'est posé sur SYMANTEC *Visual Cafe* 3.0. La justification de ce choix repose primo, sur la mise à notre disposition de la version la plus récente de ce logiciel (Version 3.0) et, secundo, sur le fait que cette dernière version supporte les dernières mises à jour du JDK de SUN, à savoir la version JDK1.2 que SUN a dernièrement renommé JDK2.0, et les puissants outils qui lui ont été adjoints.

### 4.1.3 Choix du matériel de test et de développement

Afin de mener à bien l'implantation partielle du modèle conceptuel de l'agent ARIII, nous nous sommes servis d'un certain nombre d'éléments matériels qui ont été disponibles durant la période de développement et de test. Bien que l'implantation ait été conçue à l'aide de Java pour être portable sur n'importe quel type d'architecture matérielle et n'importe quel type de système d'exploitation existant, il reste que nous nous sommes contentés, restrictions budgétaires obligent, à tester notre modèle uniquement sur des machines de type PC. Nous avons développé et effectué les tests du prototype de l'agent ARIII sur une machine possédant les caractéristiques suivantes :

- Processeur : Pentium II 300 MHz ;
- Mémoire vive (RAM) : 64 MB ;

- Carte réseau : Intel 82557 based Ethernet PCI Adapter ;
- Système d'exploitation : Windows 98.

D'autre part, afin d'évaluer la viabilité de notre prototype en le comparant à un système couramment utilisé, nous avons été amenés à mettre en place une application simulant la fonction d'un moteur de recherche basé sur l'engin *Altavista Search Engine* 3.0. Ce dernier est mis gracieusement à la disposition des chercheurs et programmeurs pour des fins de recherche et développement. La mise en place de ce système a requis l'utilisation d'un PC ayant les caractéristiques suivantes :

- Processeur : Pentium 300 MHz.
- Mémoire vive (RAM) : 256 MB ;
- Système d'exploitation : Windows NT 4.0 avec le Service Pack 5 ;
- Espace disque : 5 GB après installation ;
- Allocation mémoire virtuelle : 1 GB ;

## 4.2 Ensemble des éléments implantés du modèle conceptuel

Les travaux d'implantation de l'agent ARIII ont été concentrés sur la mise en place des principaux besoins fonctionnels et opérationnels que nous avions établis dans le chapitre précédent et qui assurent la prestation principale de l'agent à savoir la recherche d'information basée sur le profil de l'utilisateur. Les acteurs principaux de notre modèle qui sont concernés par cette implantation sont : l'utilisateur, l'administrateur et le navigateur Internet. L'agent pair, n'étant pas considéré en soit comme entité indispensable pour l'activité de la recherche d'information, sera écarté de la phase d'implantation. Le but principal de cette phase est de démontrer la faisabilité et la viabilité de notre modèle et d'offrir un prototype de l'agent ARIII qui permettra de comparer l'efficacité de sa recherche à d'autres engins fonctionnels existants.

Nous décrivons, dans ce qui suit, l'ensemble des éléments implantés de notre modèle en décrivant au fur et à mesure les classes du modèle qui sont impliquées dans cette réalisation.

***Classe des agents ARIII***

La classe des agents ARIII est évidemment la base de notre implantation. Elle permet d'instancier des agents ARIII qui offrent à travers leurs interfaces des services de recherche intelligente d'information, comme nous l'avons décrit au chapitre précédent. La Figure 4.1 fournit une description sommaire de la classe des agents ARIII. Celle-ci est implantée dans notre réalisation à l'aide de la classe java *ARIII.class*.

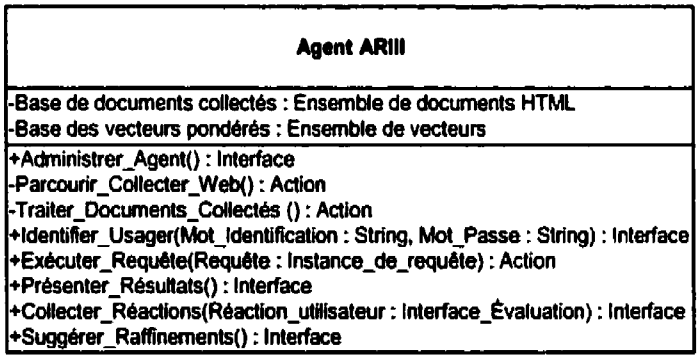


**Figure 4.1 Description UML de la classe des agents ARIII**

***Classe des usagers***

La classe des usagers correspondant à la classe java *Usagers.Class* permet d'instancier des usagers qui seront associés à une instance donnée de la classe des agents ARIII. La classe des usagers est en effet une super classe pour les classes des utilisateurs (*Utilisateur.class*) et des administrateurs (*Administrateur.class*). Ces classes offrent respectivement, à travers leurs interfaces, aux utilisateurs et aux administrateurs de l'agent la panoplie d'opérations et services définis au sein de notre modèle. La Figure 4.2 illustre la réalisation de la hiérarchie des classes usagers.

***Classe des profils***

La classe des profils permet d'instancier des profils qui seront respectivement associés à une instance particulière d'utilisateur et à une instance particulière d'agent ARIII. Leurs interfaces comportent des opérations élémentaires de définition et de manipulation d'un profil, tel que définit notre modèle conceptuel. La Figure 4.3 fournit une description sommaire de la classe des profils. Celle-ci est implantée dans notre réalisation à l'aide de la classe java *Profil.class*.

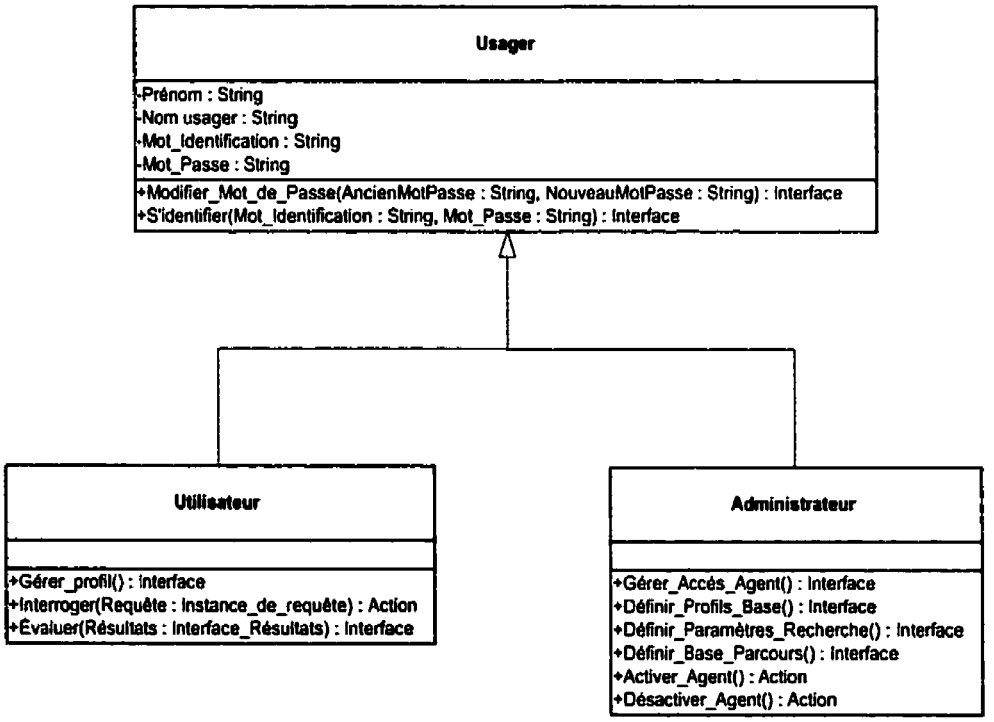


**Figure 4.2 Description UML de la hiérarchie de classes des usagers**

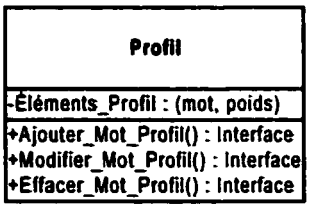


**Figure 4.3 : Description UML de la classe des profils**

***Classe des requêtes***

La classe des requêtes permet d'instancier des requêtes qui seront respectivement associées à une instance particulière d'utilisateur et à une instance particulière d'agent ARIII. Leurs interfaces comportent des opérations élémentaires de définition et de manipulation d'une requête, tel que définit notre modèle conceptuel. La Figure 4.4 fournit une description sommaire de la classe des requêtes. Celle-ci est implantée dans notre réalisation à l'aide de la classe java *Requete.class*.

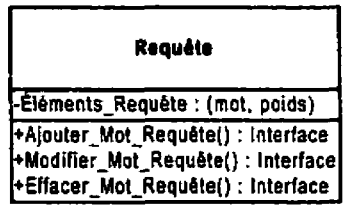


**Figure 4.4 Description UML de la classe des requêtes**

***Classe des Navigateurs Internet***

La classe des navigateurs Internet permet d'instancier des modules de navigation qui seront respectivement associés à une instance particulière d'agent ARIII. Leurs interfaces comportent des opérations élémentaires d'accès et de collecte de documents sur le WEB, tel que définit notre modèle conceptuel. Cette classe est implantée dans notre réalisation à l'aide de la classe java *NavigateurInternet.class*. Celle-ci est basée sur les classes *URLConnection* et *URL* du paquetage java *java.net* inclus dans la distribution (JDK 1.2) SUN de JAVA. La Figure 4.5 fournit une description sommaire de la relation de composition entre la classe composite des Navigateurs Internet et ses classes composantes. La Figure 4.6 résume les relations d'association entre l'ensemble des classes implantées à l'aide du formalisme UML.

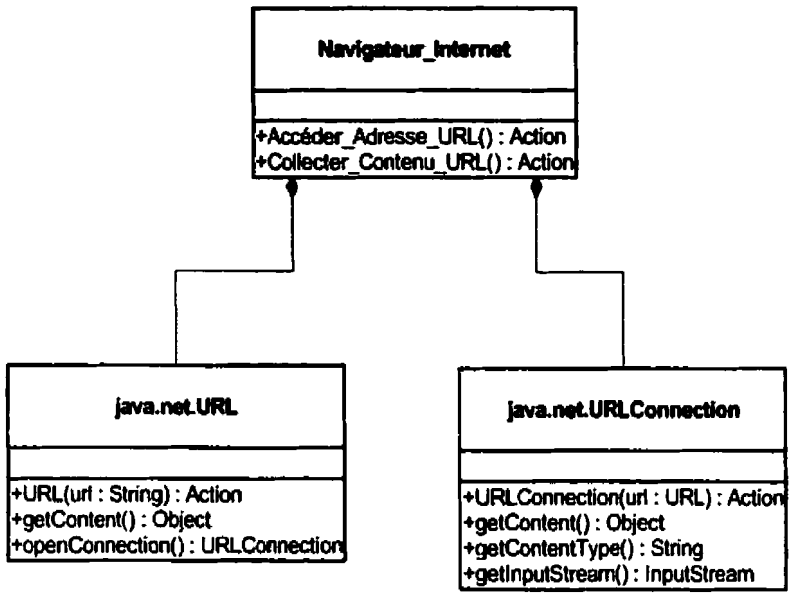


**Figure 4.5 La classe des navigateurs Internet et de ses références**

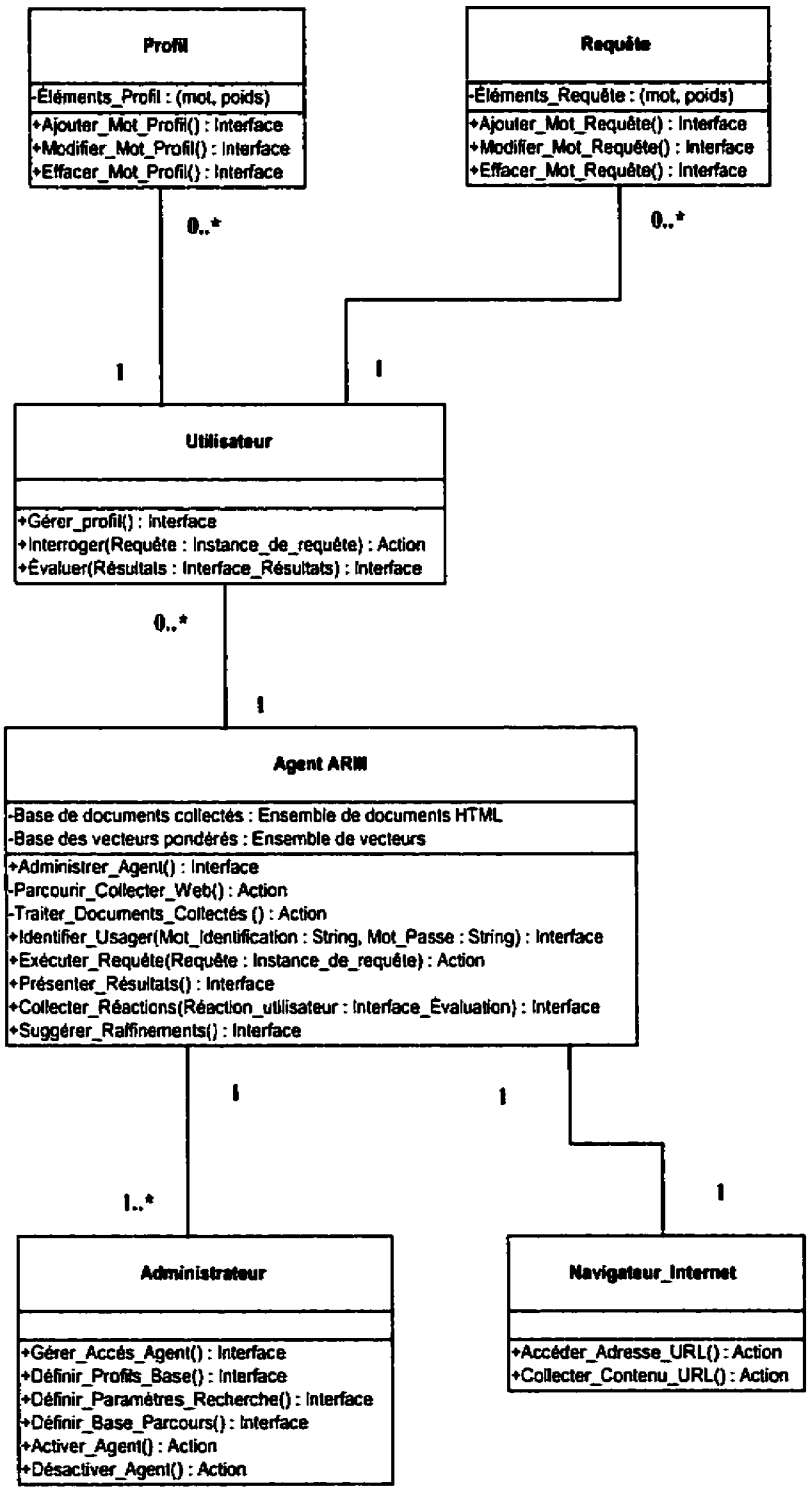


**Figure 4.6 Sommaire des classes implantées**

## 4.3 Mise en œuvre et évaluation du modèle

Dans cette section, nous mettons en œuvre notre implantation partielle du modèle conceptuel des agents ARIII en présentant quelques-unes des fonctionnalités du système implanté. Ainsi, nous allons donner lors de cette étape quelques illustrations de l'interface et des interactions à travers des scénarios d'utilisation qu'on peut appeler aussi des cas d'utilisation. Par la suite, nous présenterons une expérimentation que nous avons menée afin d'évaluer notre système implanté en le comparant à un système existant. Cette expérimentation nous permettra de situer notre outil de recherche d'information sur le WEB par rapport aux moteurs de recherche classiques.

### 4.3.1 Élaboration des scénarios d'interactions

Dans cette sous-section nous allons élaborer deux scénarios mettant à contribution les deux types d'usager administrateur et utilisateur. Nous présentons lors de ces scénarios succinctement différents services et interfaces offerts par notre implantation.

***Scénario 1 : Interaction Utilisateur / ARIII***

Dans ce scénario, nous supposons l'existence d'une instance nommée *Agent_ARIII* de la classe ARIII. Nous supposons aussi que cet agent est fonctionnel, dans le sens qu'il possède une base de documents et que ses paramètres de gestion ont été établis par l'un de ses administrateurs. Nous supposons qu'un utilisateur nommé $U_{Agent\_ARIII}$ possède les droits d'accès à l'instance *Agent_ARIII* et qu'il possède toutes les informations (identification et mot de passe) lui permettant d'accéder aux services de l'agent.

L'utilisateur $U_{Agent\_ARIII}$ doit effectuer les étapes du scénario suivant :

1. S'identifier auprès de l'agent *Agent_ARIII* en fournissant ses paramètres d'accès et accéder aux services qui lui sont octroyés.
2. Créer un nouveau profil et le valider.
3. Créer une requête d'information et la faire exécuter auprès de l'agent.
4. Visualiser et évaluer les résultats de sa requête.

*Étape 1*:

$U_{Agent\_ARIII}$ s'identifie auprès de l'agent *Agent_ARIII* en fournissant ses paramètres d'accès à travers une interface d'accès commune à tous les usagers du système, quel que soit la catégorie ou le type de ces usagers. Il s'agit donc d'une simple interface d'accès qui exigera et validera le mot d'identification (*login*) et le mot de passe (*password*) de $U_{Agent\_ARIII}$. La Figure 4.7 illustre cette interface d'accès à l'agent.

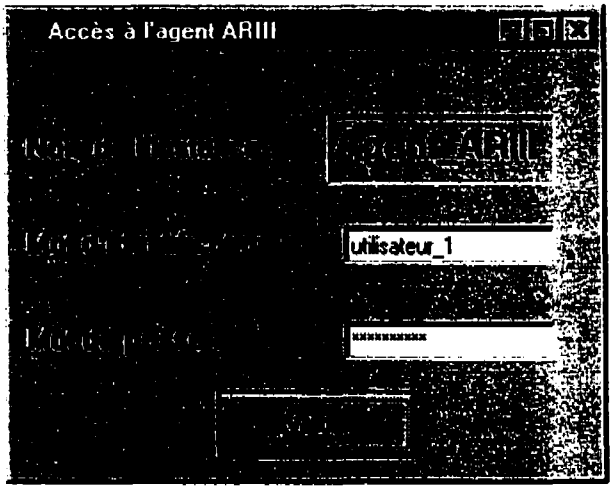


**Figure 4.7 Interface d'accès à l'agent ARIII**

*Étape 2 :*

Une fois l'accès réussi, une interface principale d'accueil sera présentée à l'utilisateur. Celle-ci lui fournit la panoplie des services auxquels il a droit. Cette interface diffère par son contenu, selon le type d'usager reconnu. Ainsi si l'usager est un administrateur, il aura accès à des services et fonctions différents de ceux d'un utilisateur. La Figure 4.8 illustre l'interface principale d'accueil d'un utilisateur. L'utilisateur est invité à choisir la fonction de gestion de profil qui lui permet d'ajouter un nouveau profil. La Figure 4.9 illustre l'interface permettant une telle création. L'utilisateur peut ajouter des mots et leurs pondération (valeur entre 1 et 5) à son profil.

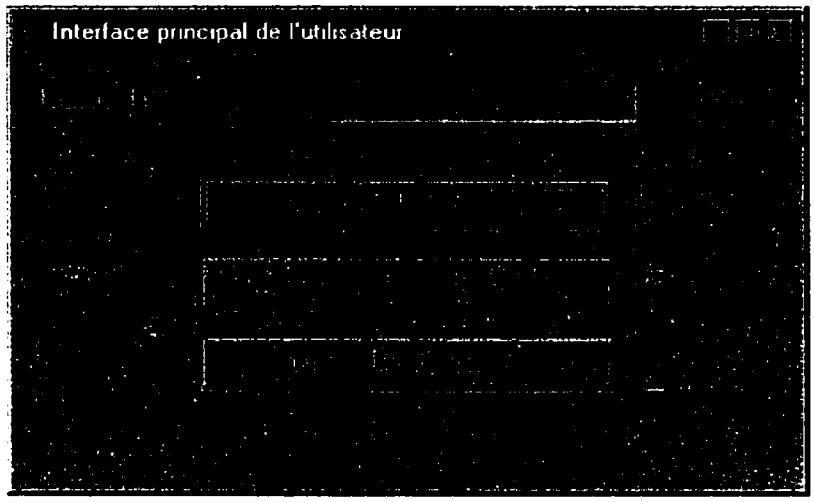


**Figure 4.8 Interface principale de l'utilisateur**

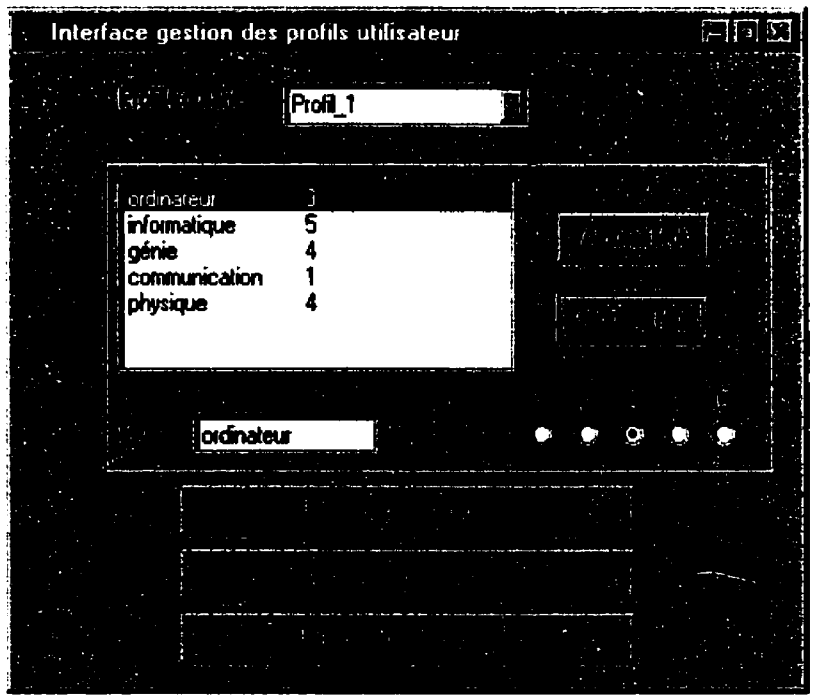


**Figure 4.9 Interface de création et de modification des profils**

*Étape 3 :*

Une fois le profil validé, l'utilisateur peut choisir à partir de l'interface principale de créer une requête. Tout comme dans le cas du profil, il est invité à saisir un certain nombre de mots et à les pondérer (valeur entre 1 et 5). À partir de la même interface, l'utilisateur pourra lancer sa recherche. La Figure 4.10 illustre la création d'une requête et le lancement de la recherche.


**Figure 4.10 Interface de définition et de lancement de requête**

*Étape 4 :*

Une fois les résultats de la requêtes obtenus, une interface de présentation et d'évaluation des résultats sera alors présentée à l'utilisateur. Celle-ci lui fournit, dans l'ordre décroissant de pertinence, les résultats de la requêtes. L'utilisateur est invité dans cette interface à réagir face à ses résultats en donnant une note de satisfaction (0 et 100 %)

pour chaque document qu'il désire évaluer. L'utilisateur peut, à l'issue de cette évaluation, relancer la recherche. La Figure 4.11 illustre l'interface de présentation des résultats quant à la Figure 4.12 elle montre l'interface d'évaluation d'un résultat.

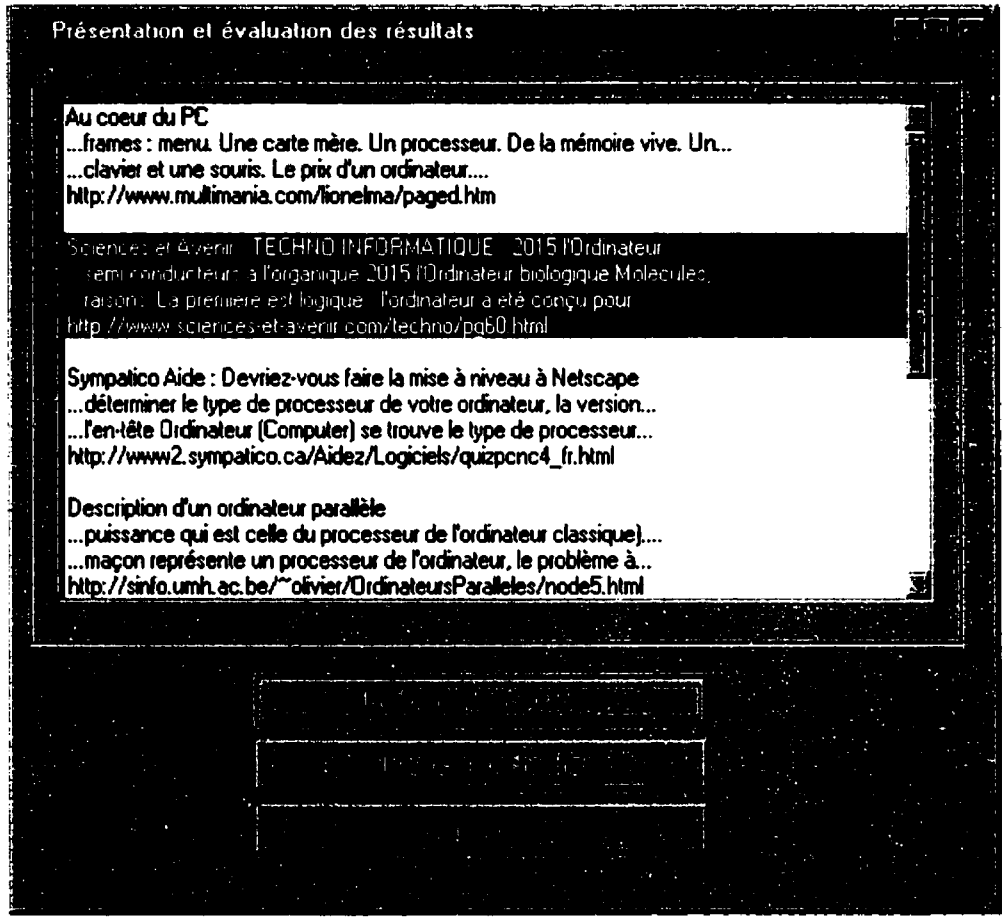


**Figure 4.11 Interface de présentation et d'évaluation des résultats**

***Scénario 2 : Interaction Administrateur /ARIII***

Dans ce scénario, nous supposons l'existence d'une instance nommée *Agent_ARIII* de la classe ARIII. Nous supposons qu'un adminitrateur nommé $ADM_{Agent\_ARIII}$ possède les droits d'accès à l'instance *Agent_ARIII* ainsi que toutes les informations (identification et mot de passe) lui permettant d'accéder aux services de l'agent.

L'administrateur $ADM_{Agent\_ARIII}$ doit effectuer les étapes du scénario suivant :

1. S'identifier auprès de l'agent *Agent_ARIII* en fournissant ses paramètres d'accès et accéder aux services qui lui sont octroyés.
2. Définir la base du parcours du WEB.
3. Définir les paramètres de la recherche de l'agent.

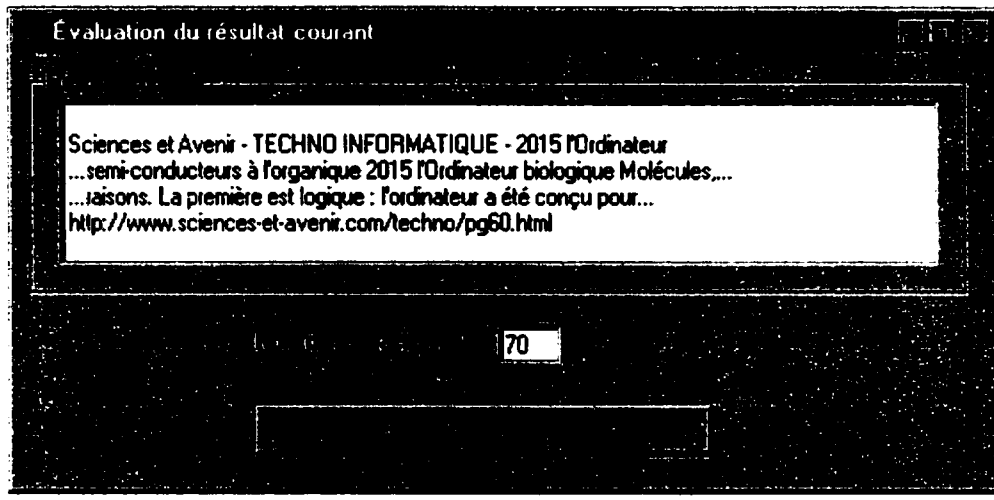


**Figure 4.12 Interface d'évaluation d'un résultat**

*Étape 1*:

$ADM_{Agent\_ARIII}$ s'identifie auprès de l'agent *Agent_ARIII* en fournissant ses paramètres d'accès (voir Scénario 1, Étape 1).

*Étape 2 :*

Une fois l'accès réussi, une interface principale d'accueil sera présentée à l'administrateur $ADM_{Agent\_ARIII}$. Celle-ci lui fournit la panoplie des services auxquels il a droit. La Figure 4.13 illustre l'interface principale d'accueil d'un administrateur. L'administrateur est invité à choisir la fonction *Définir la base du parcours* lui permet de saisir la liste des adresses URL constitueront la base du parcours et de la collecte d'informations sur Internet. La Figure 4.14 illustre l'interface associée à cette fonction.

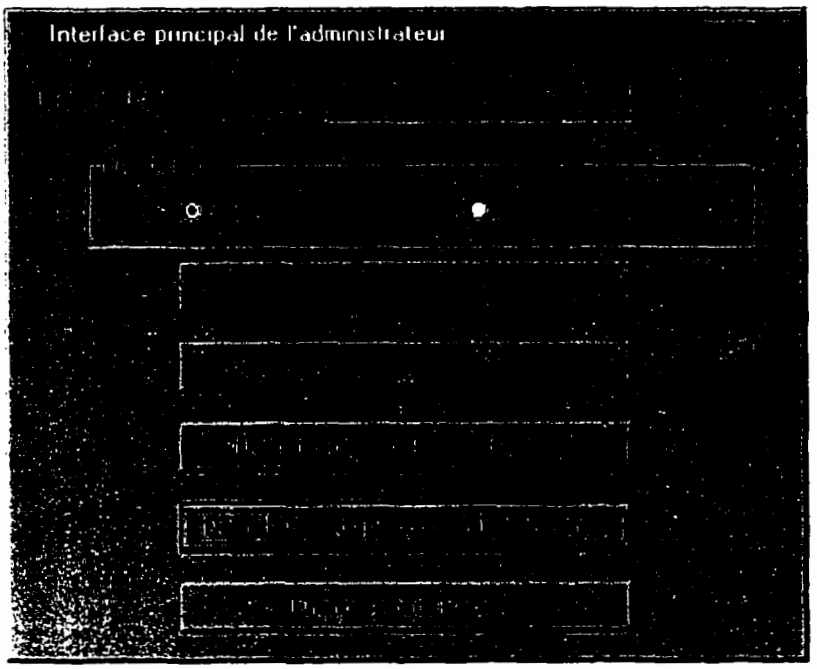


**Figure 4.13 Interface principale d'un administrateur**

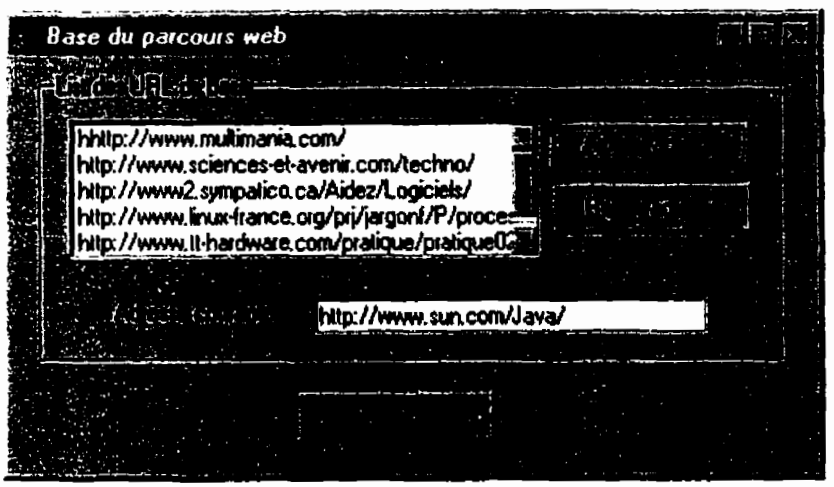


**Figure 4.14 Définition de la base du parcours**

*Étape 3 :*

Une fois la création de la base du parcours réussie, l'interface principale d'accueil sera présentée à nouveau à l'administrateur $ADM_{Agent\_ARIII}$. Celui-ci pourra donc choisir la fonction *Définir les paramètres de la recherche* (Figure 4.13). La Figure 4.15 illustre

l'interface principale qui permet à un administrateur de définir et de valider les paramètres de la recherche de l'agent.

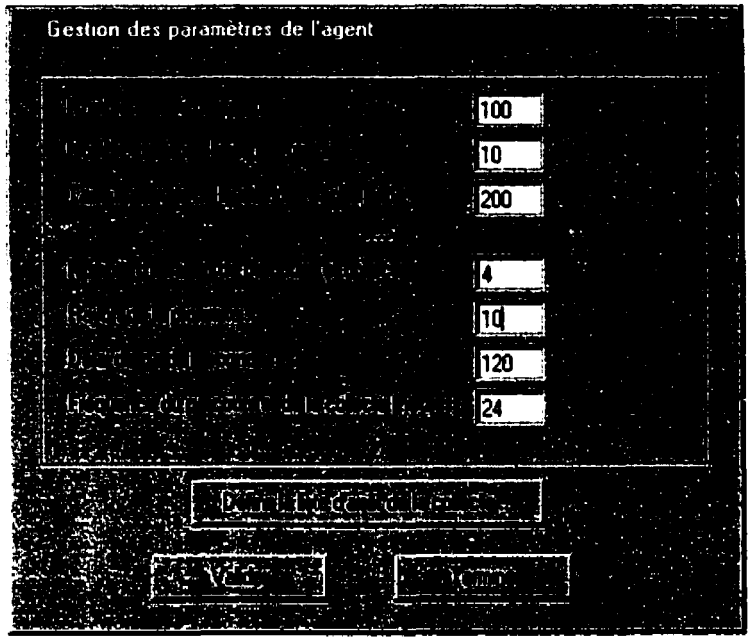


**Figure 4.15 Interface de gestion des paramètres de l'agent**

### 4.3.2 Expérimentation et évaluation

Afin d'évaluer la pertinence et la viabilité de notre modèle conceptuel, il est impératif de définir le principal critère sur lequel nous allons établir cette évaluation. Puisque l'objectif principal de notre modèle est d'améliorer la pertinence des résultats d'une recherche d'information en ordonnant ceux-ci selon un classement tenant compte du profil de l'utilisateur, il est évident que notre principal critère d'évaluation sera ce classement. Cependant, évaluer la pertinence d'un classement est une tâche complexe, puisque cette pertinence est subjective et dépend entièrement des besoins informationnels d'un utilisateur donné. Ainsi, deux utilisateurs distincts A et B, soumettant une même requête, peuvent apprécier différemment la pertinence d'un même document résultat, donc sa position dans le classement des résultats. Par conséquent, deux stratégies d'évaluation peuvent être adoptées :

1. Une expérimentation peut être menée sur un grand échantillon d'utilisateurs en collectant leurs degrés de satisfaction face aux résultats du prototype.
2. Une comparaison de deux systèmes, notre prototype et un outil de recherche existant, peut être entreprise, et une évaluation sera obtenue en collectant le degré de satisfaction d'un utilisateur donné face aux résultats obtenus dans ces deux systèmes suite à une requête donnée.

La première stratégie exige que l'expérimentation soit menée avec la participation d'un grand nombre de personnes, et sous-entend une période d'évaluation qui peut être plus ou moins longue, ce qui dépasserait le contexte de ce mémoire. La deuxième stratégie est donc celle adoptée lors de l'expérimentation que nous avons menée. Nous décrivons dans ce qui suit notre plan d'expérimentation basé sur cette stratégie de comparaison. Notons ici qu'il est impossible, dans le cadre de ce mémoire, de mener une comparaison universelle qui tient compte de tous les engins de recherche existants et de tous les cas de figures possibles et imaginables qu'un système de recherche d'information puisse rencontrer. Nous allons donc restreindre la comparaison de notre prototype à un seul moteur de recherche (*Altavista*) considéré parmi les plus utilisés, et les plus fiables sur Internet. Cette comparaison va tenir compte de trois scénarios dont le deuxième représente un cas d'utilisation typique et général des deux systèmes de recherche d'information sur Internet, alors que le premier et le troisième représentent, quant à eux, deux cas limites d'utilisation qui testeront les comportements de deux systèmes mis à compétition dans ces cas extrêmes.

*Plan de l'expérimentation*

Notre expérimentation vise à comparer notre prototype d'agent ARIII avec un système de recherche basé sur l'engin *Altavista Search Engine* 3.0.

Pour comparer les deux systèmes, il faut les amener à travailler sur un même échantillon de données. À toutes fins pratiques, nous avons réduit l'échantillon à 100 documents en format HTTP et nous avons mené une expérimentation qui s'articule autour de trois

scénarios différents. Les considérations suivantes ont été respectées dans chacun de ces scénarios :

1. La même liste d'arrêt (*stop list*) a été utilisée afin de ne pas pénaliser la performance d'un système par rapport à l'autre. Rappelons ici que la liste d'arrêt établit les termes qui seront éliminés d'office lors de l'indexation ou de l'établissement de la base de documents.
2. L'expérimentation implique un seul utilisateur donné A, ayant un profil courant $PU_A$.
3. Une même requête informationnelle est adressée aux deux systèmes. Rappelons ici que les deux systèmes possèdent chacun son propre langage d'interrogation (booléen pour *AltaVista* et mots clés pondérés pour notre prototype), et donc une formulation différente de la requête. Cependant, nous allons nous restreindre pour les fins de la comparaison aux requêtes possédant des formulations comme suit :
   - *Altavista* : Ensemble de termes séparés d'espace et donc utilisant l'opérateur booléen non explicite OU.
   - *ARIII* : Ensemble de termes ayant tous le même poids (5 par exemple).

Parmi notre échantillon de documents, nous avons identifié pour chacun des scénarios un document particulier que nous appelons désormais le document cible $DOC_{Cible}$ et qui correspond à un document considéré comme le plus pertinent pour la requête informationnelle adressée aux deux systèmes. L'existence ou l'absence de ce document cible $DOC_{Cible}$ parmi les résultats de la recherche et, lorsqu'il figure parmi les résultats, sa position dans le classement des résultats, forment le critère de comparaison et d'évaluation des deux systèmes.

Dans ce qui suit, nous allons exposer les trois différents scénarios mettant les deux systèmes en compétition et nous allons donner pour chacun de ces scénarios une interprétation des résultats obtenus.

*Scénario 1 :*

Description de la base de documents :

La base de documents est un ensemble de 100 documents extraits du WEB dont le thème commun est les mathématiques et dont le mot "Mathématiques" est présent, et relativement fréquent dans tous les documents.

Requête et profil :

La requête est composée des trois mots clés suivants : "Mathématiques", "Suite", "Fibonacci".

Dans ce premier scénario, le profil de l'utilisateur est désactivé dans notre prototype, et il n'est pas utilisé lors de la recherche.

Document cible :

Le document cible correspond à l'adresse de la page WEB suivante : "http://www.ac-nantes.fr/peda/disc/math/lgallard/doc2/fibonacci.htm". Ce document traite de l'énoncé du problème de Fibonacci. Nous le considérons comme étant le document résultat le plus pertinent de notre requête.

Résultats et interprétation :

À l'issue de la recherche effectuée sur les deux systèmes mis en compétition, nous avons obtenu les résultats suivants :

- *ARIII* : 13 documents sur 100 sont retournés à l'issue de la recherche. Le document cible est en $2^{ème}$ position dans le classement des résultats.
- *Altavista Search Engine 3.0* : 100 documents sur 100 sont retournés à l'issue de la recherche. Le document cible est en $8^{ème}$ position dans le classement des résultats.

Bien qu'il s'agisse d'une recherche n'impliquant pas le profil de l'utilisateur, il apparaît de ces résultats que notre prototype place en meilleure position le document cible. Ceci est attribuable à la nature de l'indexation utilisée dans chacun des systèmes. En fait, l'engin *Altavista* indexe les mots des documents selon leur nombre d'occurrence et, par conséquent, lors de son classement des résultats, l'engin place en premier lieu les

documents ayant la plus grande occurrence des mots de la requête. D'autre part, puisque notre prototype utilise le modèle vectoriel avec une pondération basée sur la normalisation TF*IDF, le mot "mathématiques" est éliminé de la requête (sa pondération est de 0 puisque tous les documents contiennent ce mot, cf. Figure 3.2). Ainsi, notre prototype élimine tous les documents ne contenant pas les mots "Fibonacci" et "Suite", et classe les autres documents selon la distance Euclidienne entre leurs vecteurs et le vecteur requête. Par conséquent, le classement effectué par notre prototype confère une meilleure position aux documents dont les composantes vectorielles correspondant aux mots "Suite" et Fibonacci" ont un poids proche du poids de la requête. Par ailleurs, *Altavista* n'ayant pas éliminé le mot "mathématiques" qui est présent dans tous les documents, il retourne tous les documents de la base, ce qui ne rend pas service à l'utilisateur.

*Scénario 2 :*

Description de la base de documents :

La base de documents est un ensemble de 100 documents extraits du WEB dont 50 ont un thème commun qui est "les animaux félins" et 50 autres dont le thème commun est "l'automobile".

Requête et profil :

La requête est composée du mot clé suivant : "Jaguar".

Dans ce second scénario, le profil de l'utilisateur est activé, et il est décrit comme suit :

$$PU_A = \{ (\text{"moteur"} ; 4) , (\text{"mécanique"} ; 5) \}$$

Document cible :

Le document cible correspond à l'adresse de la page WEB suivante :

" http://www.jaguar.com/fr/sdc/index.html". Ce document traite de l'histoire des voitures de marque Jaguar. Nous le considérons comme étant le document résultat le plus pertinent de notre requête.

Résultats et interprétation :

À l'issue de la recherche effectuée sur les deux systèmes mis en compétition, nous avons obtenu les résultats suivants:

- *ARIII* : 7 documents sur 100 sont retournés à l'issue de la recherche. Le document cible est en 1ère position dans le classement des résultats.
- *Altavista Search Engine 3.0* : 21 documents sur 100 sont retournés à l'issue de la recherche. Le document cible est en 17ème position dans le classement des résultats.

Il découle encore de ces résultats que notre prototype place en meilleure position le document cible. Ceci est attribuable au fait que dans ce cas le profil de l'utilisateur a été mis à contribution lors de la recherche. Ainsi, tous les vecteurs correspondant aux documents de la base dont le sujet est "les animaux félins" et qui ne contiennent pas les mots "mécanique" et "moteur" seront éloignés du vecteur requête de notre prototype. *Altavista*, ne tenant pas compte du profil de l'utilisateur, a renvoyé des documents résultats faisant alternativement partie de l'ensemble des documents sur les animaux félins et de l'ensemble des documents sur l'automobile.

*Scénario 3 :*

Description de la base de documents :

La base de documents est un ensemble de 100 documents extraits du WEB dont le thème principal est "les carburants" et dont le mot "carburant" est présent et relativement fréquent dans tous les documents.

Requête et profil :

La requête est composée du mot clé : "carburant".

Dans ce troisième scénario, le profil de l'utilisateur est activé, et il est décrit comme suit :

$$PU_A = \{ (\text{"moteur"} ; 4) , (\text{"mécanique"} ; 5) \}$$

Document cible :

Le document cible correspond à l'adresse de la page WEB suivante :

"http://perl.club-internet.fr/cgi-bin/ehmel/ehmel_search.pl?query=carburant".

Ce document traite du sujet des carburants mais il ne contient pas de référence au mot du profil : "mécanique". Nous le considérons comme étant le document résultat le plus pertinent de notre requête.

Résultats et interprétation :

À l'issue de la recherche effectuée sur les deux systèmes mis en compétition, nous avons obtenu les résultats suivants:

- *ARIII* : 58 documents sur 100 sont retournés à l'issue de la recherche. Le document cible est en 36$^{ème}$ position dans le classement des résultats.
- *Altavista Search Engine 3.0* : 100 documents sur 100 sont retournés à l'issue de la recherche. Le document cible est en 18$^{ème}$ position dans le classement des résultats.

Dans ce scénario, nous remarquons que le moteur de recherche *Altavista* est plus pertinent que notre prototype. En effet, puisque notre prototype utilise le modèle vectoriel avec une pondération basée sur la normalisation TF*IDF, l'unique mot clé de la requête, le mot "carburant", est éliminé lors de la normalisation des vecteurs. En effet, sa pondération est de 0 puisque tous les documents contiennent ce mot. Ainsi, l'exécution de la requête va se faire à partir du vecteur profil uniquement. L'ensemble des résultats sont des documents qui correspondent à une requête composée des mots clés "moteur" et "mécanique". Ceci explique le mauvais classement du document qui fait partie des résultats malgré tout, puisqu'il contient le mot "moteur". Notons que ce cas de figure est peu probable dans le cas d'une application réelle, il montre néanmoins un cas de limitation de notre prototype.

À l'issue de cette expérimentation, nous pouvons conclure que notre modèle réduit, d'une part, le nombre de résultats retournés en ciblant et filtrant uniquement les documents qui répondent à la requête posée et au profil de l'utilisateur. De plus, ce modèle améliore sans aucun doute la pertinence des résultats d'une recherche

d'information en ordonnant ceux-ci selon un classement tenant compte du profil de l'utilisateur, et donc servant au mieux les intérêts de ce dernier et le soulageant de la tâche pénible de tri manuel parmi un grand volume de résultats retournés.

# CHAPITRE 5
# CONCLUSION

Ce chapitre propose une synthèse des travaux complétés dans le cadre de ce mémoire ainsi qu'une présentation des travaux demeurant à être complétés afin d'enrichir le prototype ARIII actuel.

## 5.1 Synthèse des travaux

Les travaux présentés dans ce mémoire portent sur l'élaboration et l'implantation partielle d'un agent, nommé ARIII, pour la recherche intelligente d'information sur Internet. Ce modèle constitue en fait un assistant de recherche offrant à des utilisateurs un service de recherche d'information sur Internet visant à faciliter recherche d'information et en améliorer la pertinence des résultats.

Comme en fait foi ce mémoire, l'élaboration d'agent ARIII s'est effectué en plusieurs étapes : analyse des outils de recherche d'information existants en mettant l'accent en particulier sur le concept de bibliothèque virtuelle, présentation des agents et des systèmes multi-agents comme solution possible aux besoins d'implantation du concept de bibliothèque virtuelle, élaboration d'un modèle d'agent de recherche intelligent d'information sur Internet s'intégrant facilement dans un environnement de bibliothèque virtuelle et offrant un service d'assistance pertinente aux utilisateurs, sélection d'un sous-ensemble d'éléments pour implantation immédiate, programmation et test de ce sous-ensemble, puis analyse du modèle en comparant son comportement avec celui d'un autre système de recherche connu.

La grande force de l'agent ARIII réside dans sa capacité d'améliorer la pertinence des résultats de la recherche d'information sur Internet en utilisant, lorsqu'il est spécifié, le profil de l'utilisateur. En effet, grâce au travail de l'agent ARIII, les utilisateurs n'ont pas à se soucier de formuler des requêtes extrêmement précises et qui contiennent

explicitement toutes les caractéristiques de leurs besoins informationnels, les profils des utilisateurs auprès de l'agent seront mis en contribution à cet effet par l'agent. Ceci allège, d'une part, la tâche de formulation d'une requête auprès du système de recherche et rend, d'autre part, plus pertinents les résultats d'une recherche effectuée à partir d'une requête simplifiée.

Le modèle de l'agent ARIII intègre aussi les mécanismes nécessaires à la mise à jour d'une manière récurrente et périodique de la base de documents sur laquelle s'effectue la recherche. Ainsi, le modèle prend en compte la nature dynamique et volatile de l'information présente sur Internet.

De plus, le modèle de l'agent ARIII offre aussi les bénéfices d'une architecture ouverte, puisqu'il permet en tout temps à une instance de l'agent ARIII de s'intégrer aisément dans un environnement multi-agent qui offre à des utilisateurs une multitude de services à l'image d'une bibliothèque virtuelle. Finalement, le modèle ARIII offre la possibilité d'instancier plusieurs agents spécialisés dans différents domaines, permettant dans un environnement de collaboration entre agents pairs d'offrir, à partir d'une seule interface aux utilisateurs, des informations plus ciblées et plus spécialisées.

## 5.2 Travaux futurs

Bien que le modèle de l'agent ARIII ait été partiellement concrétisé dans ce mémoire, plusieurs travaux demeurent à être complétés afin de faire de cet agent un outil versatile et complet pour la recherche intelligente d'information sur Internet. Dans un premier temps, certains des objets implantés jusqu'à ce jour devront être complétés et testés en profondeur sur des échantillons plus volumineux. C'est le cas notamment des structures de données et processus prévus pour stocker et manipuler les documents formant la base de l'agent. Dans un deuxième temps, les objets prévus dans le modèle original mais toujours absents dans l'environnement actuel devront être mis en place. Il faudra donc développer des modules supportant des langages standards de communication et d'échange de connaissances inter-agents tels que KQML ou KIF, permettant ainsi aux

agents ARIII de s'intégrer aisément et d'évoluer dans des environnements multi-agents complexes.

Finalement, il serait souhaitable que plusieurs éléments non abordés dans le cadre de ce mémoire mais pouvant grandement contribuer à l'amélioration du modèle ARIII soient explorés : élaboration d'un langage d'interrogation et de requête plus sophistiqué et adapté au niveau de compétence de l'utilisateur, analyse et intégration de mécanismes permettant de parcourir et collecter les informations à partir de sources hétérogènes telles que les bases de données, les bases de connaissances, etc., intégration de la notion de requêtes persistantes qui demeurent en mise à jour perpétuelle (exemple : les requêtes sur des titres en bourse, ou aussi sur l'état de la météo) et dont les résultats sont acheminés périodiquement aux utilisateurs.

# BIBLIOGRAPHIE

ALAIN, H. (1996). Les systèmes multi-agents : mythes ou réalités. Hermes, Paris.

ALEXA (2000). Alexa Insider's Page. http://insider.alexa.com .

ALIS (1996). La série de normes ISO 8859. Internet Society et Alis Technologies. http://babel.alis.com:8080/codage/iso8859/jeuxiso.htm

ATKINS, D., BIRMINGHAM, W., EDMUND, H., ERIC, J. et MICHAEL, P. (1996). Toward Inquiry Based Education Through Interacting Software Agents. University of Michigan Digital Library, Mai.

BALABANOVIC, M., SHOHAM, Y. ET YUN, Y. (1995). An Adaptive Agent for Automated Web Browsing. Stanford University Digital Library Project, Working Paper SIDL-WP-1995-0023. Stanfonrd, CA 94305, USA.

BALABANOVIC, M. ET SHOHAM, Y. (1995). Learning Information Retrieval Agents : Experiments with Automated Web Browsing. Department of Computer Science, Stanford University, Stanford, CA 94305.

BENSLIMANE, A., FEKI, T. et GOBLET, X. (1993). Conception de systèmes multi-agents : application aux tuteurs intelligents. Actes du colloque international en informatique cognitive des organisations, Montréal.

BIRMINGHAM, W., EDMUND, H., MULLEN, T. et WELLMAN, M. (1995). The Distributed Agent Architecture of the University of Michigan Digital Library UMDL. Artificial Intelligence Laboratory, University of Michigan. Ann Arbor, MI 48105-2110.

BOLES, D., DREGER, M. et GROBJOHANN, K. (1996). MeDoc Information Broker, Harnessing the information in Literature and Full Text Databases. German Ministry for Education, Science, Research and Technology, Septembre.

BOOCH, G. (1996). Object Solutions, Addison-Wesley.

BUCKLEY, C. (1985). Implementation of the SMART Information Retrieval Retrieval [sic] System. Technical Report 85-686, Cornell University.

CAGLAYAN A. et HARRISON C. (1998). Les agents : Applications bureautiques, Internet et intranet. InterÉditions, Masson, Paris.

CAO, W., BIAN, C. et HARTVIGSEN, G. (1997). ViSe2 : An Agent Based Expert Consulting System with Efficient Cooperation. Department of Computer Science Institute of Mathematical and Physical Sciences, University of Tromso, Norway.

CHAIB-DRAA, B., MOULIN, B., MANDIAU, R., MILLOT, P., (1996). Chapter 1 - Trends in Distributed Artificial Intelligence, Foundations of Distributed Artificial Intelligence, G. M. P. O'Hare and N. R. Jennings, John Wiley & Sons Inc., pp. 3-55.

CHARKRABARTI S. et. al. (1999). Recherche intelligente sur Internet. Pour La Science, n°262.

DESFRAY (1996). Modélisation par objets, Masson, 1996.

ETZIONI O. et WELD S. D. (1995). Intelligent Agents on the Internet: Fact, Fiction, and Forecast. IEEE Expert/Intelligent Systems & Their Applications Vol. 10, No. 4, August 1995.

FERBER, J. (1995). Les systèmes multi-agents : vers une intelligence collective, InterEditions.

FERBER, J. (1997). Les systèmes multi-agents : un aperçu general. Technique et science informatiques. Vol. 16 - n° 8/1997.

FININ, T. (1993). KQML, a Language and Protocol for Knowledge and Information Exchange. Technical Report Cs-94-02, Computer Science Department, University of Maryland, UMBC.

FIPA (1996). Foundation for Intelligent Physical Agents. http://www.cselt.it/fipa/

FONER, L. (1993). What is an agent, anyway? A Sociological Case Study, Agents Memo 93-01, MIT Media Lad, Cambridge, MA, 1993.

GRIFFIN S. M. (2000). Digital Libraries Initiative. http://www.dli2.nsf.gov/dlione.

HARRISON, C., CHESS, D., KERSHENBAUM, A. (1995) Mobile Agents : Are they a good idea ?, IBM Research Report, http://www.research.ibm.com/xw-d953-mobag-ps.

ISAME (2000). Cf. PELLETIER, S-J., PIERRE, S. et HOANG, H. H. (2000).

JAULENT, P. (1990). Génie logiciel : les méthodes. SADT, SA, E-A, SA-RT, SREM, SYS-P-O, OOD, HOOD, Éditions Armand Colin.

KNOBLOCK, C. A. et ARENS Y. (1994). An Architecture for Information Retrieval. AAAI Symposium on Software Agents, march, Stanford University, p. 49-56.

KORFHAGE, R. R. (1984). Query Enhancement by User Profiles. Research and development in information retrieval pages 110-122. Cambridge University Press, 1984. ISBN 0-521-26865-6.

KORFHAGE, R. R. (1988). Information Retrieval in the Presence of Reference Points, Part1. School of Library and Information Science, University of Pittsburgh, LIS001/IS88001, Pittsburgh PA 15260, janvier.

KORFHAGE, R. R. (1991). Information Retrieval in the Presence of Reference Points, Part2. School of Library and Information Science, University of Pittsburgh, LIS034/IS91002, Pittsburgh PA 15260, mars.

KORFHAGE R. R. (1997). Information Storage and Retrieval. Wiley Computer Publishing. New York. Library of Congress Cataloging-in-publication Data, ISBN 0-471-14338-3.

KULYUKIN, V., HAMMOND, K. et BURKE, R. (1996). Automated Analysis of Structured On-line Documents. In AAAI Workshop on Internet-based Information Systems, pp. 78-86. AAAI, 1996.

LABROU, Y. et FININ, T. (1997). A Proposal for a new KQML Specification. Computer Science and Electrical Enginering Department (CSEE), University of Maryland Baltimore County (UMBC), février.

LANDER, S. et LESSER, V. (1994). Sharing Meta-Information to Guide Cooperative Search Among Heterogeneous Reusable Agents. Technical Report 94-48, University of Massachusetts, Amherst, Etats-Unis.

LIRA (1995). Cf. BALABANOVIC, M. et SHOHAM, Y. (1995).

MAES, P. (1994). Agents that Reduce Work and Information Overload. MIT Media Laboratory, Cambridge, CACM-94.

MAYFIELD, J., LABROU, Y. et FININ, T. (1995). Evaluation of KQML as an Agent Communication Language. Workshop on Agents Theories, Languages and Architectures, IJCAI '95, août, Montréal, Québec, p. 282-291.

MeDoc (1996). Cf. BOLES, D., DREGER, M. et GROBJOHANN, K. (1996).

MULLER, P. A., GAERTNER, N. (2000). Modélisation Objet avec UML, 2° Édition, Eyrolles.

MURCH, R. et JOHNSON, T. (1998). Intelligent software agents, Jeffrey Pepper, Prentice-Hall PTR, New Jersey.

MYAENG, S. et KORFHAGE, R. R. (1990). Integration of user profiles : models and experiments in information retrieval. Information Processing and Management. 26(6) : 719-738, 1990. ISSN 0306-4573, avril.

NWANA, H. (1996). Software Agents : An Overview. Intelligent Systems Research, Advanced Apllication & Technology Department, BT Laboratories, Martlesham Heath, Cambridge University press, Vol. 11 No. 3, pp. 205-244, novembre.

PAEPCKE, A., CHANG, C., GARCIA-MOLINA, H. et WINOGRAD, T. (1998) Interoperability for digital libraries worldwide, Communication of the ACM, vol. 41, n°4, 33-43.

PAZZANI, M., MURAMATSU, J. et BILLSUS, D. (1997). Syskill and Webert : Identifying interesting web sites. Department of Information and Computer Science, University of California, Irvine, CA 92717.

PAZZANI, M., NGUYEN, L. et MANTIK, S. (1995). Learning from hotlists and coldlists : Towards a WWW information filtering and seeking agent. Department of Information and computer Science, University of California, Irvine, CA 92717.

PELLETIER, S-J., PIERRE, S. et HOANG, H. H. (2000) Une architecture multi-agent de recherche d'information. INFOR Vol. 38, n° 2, May 2000.

PIERRE, S. et HOTTE, R. (1996). Vers un modele intégré de support au télé-apprentissage coopératif. Annales de Télécommunications, Vol. 51, No 5-6 (mai), pp. 272-287.

RUMBAUGH, J. et. al. (1991). Object Oriented Modeling and Design, Prentice Hall.

SALTON, G. et. al. (1974). A Vector Space Model for Text Indexing. TR 74-218, Department of Computer Science, Cornell University, 1974.

SALTON, G. et MCGILL, M. (1983). Introduction to Modern Information Retrieval. Computer Science Series, McGraw Hill Book Company, ISBN 0-07-054484-0, AACR2.

SAMIER, H. et SANDOVAL, V. (1998). La recherche intelligente sur l'Internet. Hermes, Paris.

SINGH, M. P. (1993). Declarative Representations of Multiagent Systems. IEEE Transactions on Knowledge and Data Engineering, 5(5), octobre, p. 721-740.

SULLIVAN, D. (1999). Father of Alta Vista Resigns. The Search Engine Report Newsletter, May 4. Danny Sullivan Editor.

SYSKILL (1997). Cf. PAZZANI, M., MURAMATSU, J. et BILLSUS, D. (1997).

UMDL (1995). Cf. BIRMINGHAM, W., EDMUND, H., MULLEN, T. et WELLMAN, M. (1995).

ViSe (1997). Cf. CAO, W., BIAN, C. et HARTVIGSEN, G. (1997).

WOOLDRIDGE, M. et JENNINGS, N. R. (1994). Intelligent Agents : Theory and Practice. Knowledge Engineering Review, 10 (2), 1994.